

[illegible]

```
SSSSSSSS MM MM GGGGGGGG MM MM IIIIII NN NN
SSSSSSSS MM MM GGGGGGGG MM MM IIIIII NN NN
SS MM MM MM GG GG MM MM III NN NN
SS MM MM MM GG GG MM MM III NN NN
SSSSSS MM MM MM GG GG MM MM III NNNN NN
SSSSSS MM MM MM GG GG MM MM III NNNN NN
SS MM MM MM GG GG MM MM III NN NN
SS MM MM MM GG GG MM MM III NN NN
SS MM MM MM GG GG MM MM III NN NN
SSSSSSSS MM MM GGGGGG MM MM IIIIII NN NN
SSSSSSSS MM MM GGGGGG MM MM IIIIII NN NN
```

```
LL IIIIII SSSSSSSS
LL IIIIII SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS
```

```
0001 0 %TITLE 'Minimal update calculation'
0002 0 MODULE SMG$MIN (
0003 0 IDENT = '1-016' ! File: SMGMIN.B32 Edit:STAN1016
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
```

```

31 0030 1 1++
32 0031 1 1 FACILITY: Screen Management
33 0032 1 1
34 0033 1 1 ABSTRACT:
35 0034 1 1
36 0035 1 1 This module contains routines which inspect two screen
37 0036 1 1 representations and calculate the near-minimal sequence of
38 0037 1 1 terminal commands to change the current contents of the screen
39 0038 1 1 to the new representation of the screen.
40 0039 1 1
41 0040 1 1 ENVIRONMENT: User mode, SMG package.
42 0041 1 1
43 0042 1 1 AUTHOR: Stanley Rabinowitz, CREATION DATE: 1-May-1983.
44 0043 1 1 FIND_MIN_CURSOR_POS is by RKR.
45 0044 1 1
46 0045 1 1 MODIFIED BY:
47 0046 1 1
48 0047 1 1 1-016 - STAN 6-Jun-1984. Change error messages in MSG$SET_PHYSICAL_CURSOR.
49 0048 1 1 1-001 - STAN, 1-May-1983. Initial version, mimicked SCRMIN.B32.
50 0049 1 1 --

```



```
.. 52 0050 1 ZSBTTL 'Declarations'
.. 53 0051 1
.. 54 0052 1 SWITCHES:
.. 55 0053 1
.. 56 0054 1 NONE
.. 57 0055 1
.. 58 0056 1 LINKAGES:
.. 59 0057 1
.. 60 0058 1 NONE
.. 61 0059 1
.. 62 0060 1 TABLE OF CONTENTS:
.. 63 0061 1
.. 64 0062 1
.. 65 0063 1 FORWARD ROUTINE
.. 66 0064 1
.. 67 0065 1 SMG$SET_PHYSICAL_CURSOR,      ! Move physical cursor on screen
.. 68 0066 1 SMG$$OUTPUT_MINIMAL_UPDATE, ! Output minimal update sequence
.. 69 0067 1 SMG$$FIND_MIN_CURSOR_POS,   ! Output minimum cursor sequence
.. 70 0068 1 SMG$UPDATE_PHYSICAL_CURSOR, ! Update physical cursor position
.. 71 0069 1 ERASE_LINE,                ! Erase to end-of-line
.. 72 0070 1 SET_CURSOR;                ! Generate general set-cursor
.. 73 0071 1                             ! positioning sequence.
.. 74 0072 1
.. 75 0073 1
.. 76 0074 1 INCLUDE FILES
.. 77 0075 1
.. 78 0076 1
.. 79 0077 1 REQUIRE 'RTLIN:SMGPROLOG';   ! defines psects, macros, structures,
.. 80 0155 1                             ! & terminal symbols
.. 81 0156 1 REQUIRE 'RTLIN:STRLNK.REQ';  ! JSB linkages
.. 82 0341 1
.. 83 0342 1
.. 84 0343 1 EXTERNAL REFERENCES
.. 85 0344 1
.. 86 0345 1
.. 87 0346 1 EXTERNAL ROUTINE
.. 88 0347 1
.. 89 0348 1 SMG$$OUTPUT;
.. 90 0349 1
.. 91 0350 1 !+
.. 92 0351 1 $OUTPUT_STRING
.. 93 0352 1 -----
.. 94 0353 1
.. 95 0354 1
.. 96 0355 1 MACRO
.. 97 0356 1
.. 98 M 0357 1 $OUTPUT_STRING(LEN,ADDR,ATTR) =
.. 99 M 0358 1
.. 100 M 0359 1 BEGIN
.. 101 M 0360 1 EXTERNAL ROUTINE SMG$$PUT_SCREEN;
.. 102 M 0361 1 LOCAL STATUS;
.. 103 M 0362 1 STATUS=SMG$$PUT_SCREEN(PBCB,LEN,ADDR,0,0,ATTR);
.. 104 M 0363 1 IF NOT .STATUS THEN RETURN .STATUS
.. 105 M 0364 1 END
.. 106 0365 1 X;
.. 107 0366 1
.. 108 0367 1 !+
```

```
.. 109      0368 1 | $L
.. 110      0369 1 | --
.. 111      0370 1 | Macro $L linearizes a two dimensional subscript formed by a 1-based
.. 112      0371 1 | row number and a 1-based column number, into a single 0-based
.. 113      0372 1 | subscript.
.. 114      0373 1 | --
.. 115      0374 1 |
.. 116      0375 1 | MACRO
.. 117      0376 1 |
.. 118      M 0377 1 |     $L (ROW_NUMBER, COLUMN_NUMBER) =
.. 119      0378 1 |     (ROW_NUMBER-1)*.NUM_COLS + COLUMN_NUMBER -1 %;
.. 120      0379 1 |
.. 121      0380 1 | +
.. 122      0381 1 | $MAKE_ROW_COL
.. 123      0382 1 | -----
.. 124      0383 1 | Macro $MAKE_ROW_COL takes as an input a 0-based linear index into
.. 125      0384 1 | and array and converts it into a 1-based row and 1-based column
.. 126      0385 1 | form. INDEX needs to be re-expressed as a quadword for use in the
.. 127      0386 1 | EDIV instruction.
.. 128      0387 1 | --
.. 129      0388 1 |
.. 130      0389 1 | MACRO
.. 131      0390 1 |
.. 132      M 0391 1 |     $MAKE_ROW_COL ( INDEX, ROW_NUMBER, COLUMN_NUMBER) =
.. 133      M 0392 1 |     BEGIN-      ! MAKE_ROW_COL
.. 134      M 0393 1 |     BUILTIN
.. 135      M 0394 1 |     EDIV;
.. 136      M 0395 1 |     LOCAL
.. 137      M 0396 1 |     WIDTH,
.. 138      M 0397 1 |     LOCAL_INDEX : VECTOR [2, LONG];
.. 139      M 0398 1 |     LOCAL_INDEX [1] = 0; ! Second longword is always 0
.. 140      M 0399 1 |     LOCAL_INDEX [0] = .INDEX;
.. 141      M 0400 1 |     WIDTH=.NUM_COLS; ! Store width in longword
.. 142      M 0401 1 |
.. 143      M 0402 1 |     EDIV ( WIDTH, LOCAL_INDEX, ROW_NUMBER, COLUMN_NUMBER);
.. 144      M 0403 1 |     ROW_NUMBER = .ROW_NUMBER + 1;
.. 145      M 0404 1 |     COLUMN_NUMBER = .COLUMN_NUMBER + 1;
.. 146      M 0405 1 |     END;      ! MAKE_ROW_COL
.. 147      0406 1 |     %;
```

Minimal update calculation  
SMG\$\$OUTPUT\_MINIMAL\_UPDATE - Calculate minimum

N 3  
16-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGMIN.B32:1

Page 5  
(4)

```

149      0407 1 %SBTTL 'SMG$$OUTPUT MINIMAL UPDATE - Calculate minimum update sequence'
150      0408 1 GLOBAL ROUTINE SMG$$OUTPUT_MINIMAL_UPDATE (P_PBCB) =
151      0409 1 ++
152      0410 1 FUNCTIONAL DESCRIPTION:
153      0411 1
154      0412 1 This routine compares CURR_TEXT and CURR_ATTR (which reflect
155      0413 1 what is currently on the screen), with NEW_TEXT and NEW_ATTR
156      0414 1 (which reflect what should be on the screen) and calculates a
157      0415 1 sequences of characters which when output to the screen changes
158      0416 1 the current screen contents to reflect the new (desired) screen
159      0417 1 contents. These characters are actually output to the screen.
160      0418 1
161      0419 1 CALLING SEQUENCE:
162      0420 1
163      0421 1 ret_status.wlc.v = SMG$$MINIMUM_UPDATE ( P_PBCB.rab.r)
164      0422 1
165      0423 1 FORMAL PARAMETERS:
166      0424 1
167      0425 1 P_PBCB,rab.r Address of pasteboard control block
168      0426 1
169      0427 1 IMPLICIT INPUTS:
170      0428 1
171      0429 1 Contents of PBCB and WCB
172      0430 1
173      0431 1 IMPLICIT OUTPUTS:
174      0432 1
175      0433 1 Internal buffers change.
176      0434 1
177      0435 1 COMPLETION STATUS:
178      0436 1
179      0437 1 $$$_NORMAL Normal successful completion
180      0438 1
181      0439 1 SIDE EFFECTS:
182      0440 1
183      0441 1 NONE
184      0442 1 --

```



```
186 0443 2 BEGIN
187 0444 2
188 0445 2 BUILTIN
189 0446 2
190 0447 2 CMPC3;
191 0448 2
192 0449 2 BIND
193 0450 2
194 0451 2 PBCB = .P PBCB : BLOCK[,BYTE],
195 0452 2 WCB = .PBCB[PBCB_A_WCB] : BLOCK[,BYTE],
196 0453 2 NUM_ROWS = WCB[WCB_W_NO_ROWS] : WORD,
197 0454 2 NUM_COLS = WCB[WCB_W_NO_COLS] : WORD,
198 0455 2 CUR_TEXT = .WCB[WCB_A_SCR_TEXT_BUF] : VECTOR[,BYTE],
199 0456 2 CUR_ATTR = .WCB[WCB_A_SCR_ATTR_BUF] : VECTOR[,BYTE],
200 0457 2 NEW_TEXT = .WCB[WCB_A_TEXT_BUF] : VECTOR[,BYTE],
201 0458 2 NEW_ATTR = .WCB[WCB_A_ATTR_BUF] : VECTOR[,BYTE],
202 0459 2 NEW_LCV = .WCB[WCB_A_LINE_CHAR] : VECTOR[,BYTE],
203 0460 2 CUR_LCV = .WCB[WCB_A_SCR_LINE_CHAR] : VECTOR[,BYTE],
204 0461 2 OLD_CURSOR_ROW = WCB[WCB_W_OLD_CUR_ROW] : WORD,
205 0462 2 OLD_CURSOR_COL = WCB[WCB_W_OLD_CUR_COL] : WORD,
206 0463 2 NEW_CURSOR_ROW = WCB[WCB_W_CURR_CUR_ROW] : WORD,
207 0464 2 NEW_CURSOR_COL = WCB[WCB_W_CURR_CUR_COL] : WORD,
208 0465 2 SIZE = WCB[WCB_L_BUFSIZE] : Size of buffers
209 0466 2 FIRST_ROW = PBCB[PBCB_W_FIRST_CHANGED_ROW] : WORD,
210 0467 2 LAST_ROW = PBCB[PBCB_W_LAST_CHANGED_ROW] : WORD,
211 0468 2 FIRST_COL = PBCB[PBCB_W_FIRST_CHANGED_COL] : WORD,
212 0469 2 LAST_COL = PBCB[PBCB_W_LAST_CHANGED_COL] : WORD,
213 0470 2 TERM_TYPE = PBCB[PBCB_B_DEVTTYPE] : BYTE;
214 0471 2
215 0472 2 LOCAL
216 0473 2
217 0474 2 STATUS, : Status to return to caller
218 0475 2 INDEX, : Working index into the buffers
219 0476 2 ROW, : Working row number
220 0477 2 COL, : Working column number
221 0478 2 LEN, : local length
222 0479 2 ADJUSTED_COL, : Wide line adjusted column number
223 0480 2 CUR_TEXT_PTR : REF VECTOR [,BYTE], : Current pointer into
224 0481 2 : current text buffer
225 0482 2 CUR_ATTR_PTR : REF VECTOR [,BYTE], : Current pointer into
226 0483 2 : current attribute buffer
227 0484 2 NEW_TEXT_PTR : REF VECTOR [,BYTE], : Current pointer into new
228 0485 2 : text buffer
229 0486 2 NEW_ATTR_PTR : REF VECTOR [,BYTE], : Current pointer into new
230 0487 2 : attribute buffer
231 0488 2 END_ROW_INDEX, : Index to last character in current row
232 0489 2 RENDITION, : local rendition
233 0490 2 FINAL_INDEX, : local index representing end of a changed sequence
234 0491 2 CURSOR_ROW, : Current cursor row
235 0492 2 CURSOR_COL, : Current cursor column
236 0493 2
237 0494 2 NEW_CHARS_LEFT,
238 0495 2 CHARS_LEFT; : Number of characters left to be inspected.
239 0496 2 : Starts out equal to number of characters
240 0497 2 : in the four buffers.
```



```
242 0498 2 !+
243 0499 2 ! If CTRL/O was typed previously, some QIO has returned with
244 0500 2 ! that success status and our CTRL/O bit is set. We don't
245 0501 2 ! really know what the screen looks like anymore, so we
246 0502 2 ! clear out the screen buffer.
247 0503 2 !-
248 0504 2 !-
249 0505 2 IF .PBCB[PBCB_V_CONTRLO]
250 0506 2 THEN BEGIN ! Clear screen buffer
251 0507 2     CH$FILL(0,SIZE,CUR_TEXT);
252 0508 2     FIRST_ROW=1;
253 0509 2     FIRST_COL=1;
254 0510 2     LAST_ROW=.NUM_ROWS;
255 0511 2     LAST_COL=.NUM_COLS;
256 0512 2     PBCB[PBCB_V_CONTRLO]=0
257 0513 2     END; ! Clear screen buffer
258 0514 2 !-
259 0515 2 !+
260 0516 2 ! Initialize our working pointers into the buffers.
261 0517 2 ! For now: we invalidate the initial cursor position
262 0518 2 ! to force the first update to use full cursor addressing.
263 0519 2 !-
264 0520 2 !-
265 0521 2 !CURSOR_ROW = .OLD_CURSOR_ROW;
266 0522 2 !CURSOR_COL = .OLD_CURSOR_COL;
267 0523 2 !-
268 0524 2 CURSOR_ROW=0;
269 0525 2 CURSOR_COL=0;
270 0526 2 !-
271 0527 2 INCR ROW FROM .FIRST_ROW TO .LAST_ROW DO
272 0528 2 BEGIN ! Scan row .ROW
273 0529 2     LOCAL PTEXT,PATR;
274 0530 2     LOCAL BLANK_COL;
275 0531 2     LOCAL PRE_PTR_IN_ROW; ! Pointer position just before first character
276 0532 2     ! in this row
277 0533 2     CUR_TEXT_PTR = CUR_TEXT+(.ROW-1)*.NUM_COLS;
278 0534 2     CUR_ATTR_PTR = CUR_ATTR+(.ROW-1)*.NUM_COLS;
279 0535 2     NEW_TEXT_PTR = NEW_TEXT+(.ROW-1)*.NUM_COLS;
280 0536 2     NEW_ATTR_PTR = NEW_ATTR+(.ROW-1)*.NUM_COLS;
281 0537 2 !-
282 0538 2 IF .NEW_LCV[.ROW] EQL 0
283 0539 2 THEN
284 0540 2     CHARS_LEFT=.NUM_COLS
285 0541 2 ELSE
286 0542 2     CHARS_LEFT=.NUM_COLS/2;
287 0543 2 ! CHARS_LEFT=.NUM_COLS;
288 0544 2 PRE_PTR_IN_ROW=.CUR_TEXT_PTR-1;
289 0545 2 !-
290 0546 2 !+
291 0547 2 ! See if the characteristics of this line must change.
292 0548 2 !-
293 0549 2 !-
294 0550 2 IF .CUR_LCV[.ROW] NEQ .NEW_LCV[.ROW]
295 0551 2 THEN
296 0552 2     BEGIN ! Change line characteristics
297 0553 2     LOCAL BUFFER : VECTOR[SMG$K_LONGEST_SEQUENCE,BYTE],
298 0554 2
```

```
299      BUFLN;
300      0556 4
301      0557 4      EXTERNAL ROUTINE
302      0558 4
303      0559 4      SMG$OUTPUT;
304      0560 4
305      0561 4      +
306      0562 4      | Move to the desired row.
307      0563 4      -
308      0564 4
309      0565 4      SMG$FIND_MIN_CURSOR_POS ( PBCB,      | Pasteboard Control block
310      0566 4      .CURSOR_ROW,      | Current row
311      0567 4      .CURSOR_COL,      | Current column
312      0568 4      .ROW,      | Desired row
313      0569 4      1);      | Desired column
314      0570 4
315      0571 4      +
316      0572 4      | Update our record of where we are on screen.
317      0573 4      -
318      0574 4
319      0575 4      CURSOR_ROW = .ROW ;
320      0576 4      CURSOR_COL = 1 ;
321      0577 4
322      0578 4      BUFLN=0;
323      0579 4
324      0580 4      +
325      0581 4      | Get escape sequence to change the line characteristics.
326      0582 4      -
327      0583 4
328      0584 4      SELECTONE .NEW_LCV[.ROW] OF
329      0585 4      SET
330      0586 4      [LINE_K_WIDE]:      $SMG$GET_TERM_DATA(DOUBLE_WIDE);
331      0587 4      [LINE_K_UPPER_HIGH]: $SMG$GET_TERM_DATA(DOUBLE_HIGH_TOP);
332      0588 4      [LINE_K_LOWER_HIGH]: $SMG$GET_TERM_DATA(DOUBLE_HIGH_BOTTOM);
333      0589 5      [LINE_K_NORMAL]:      $SMG$GET_TERM_DATA(SINGLE_HIGH)
334      0590 4      TES;
335      0591 4
336      0592 4      +
337      0593 4      | Output it.
338      0594 4      -
339      0595 4
340      0596 4      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
341      0597 5      THEN BEGIN
342      0598      STATUS=SMG$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
343      0599      .PBCB[PBCB_A_CAP_BUFFER]);
344      0600      IF NOT .STATUS THEN RETURN .STATUS
345      0601      END
346      0602 5
347      0603 5      END;      ! Change line characteristics
348      0604 5
349      0605 5      +
350      0606 5      | Scan backwards looking for the largest sequence of trailing spaces.
351      0607 5      | Set BLANK_COL to the column number of the start of such a suffix.
352      0608 5      -
353      0609 5
354      0610 5      BLANK_COL=.NUM_COLS+1;
355      0611 5      PTEXT=NEW_TEXT$.ROW*.NUM_COLS;
```

```
356 0612 PATTR=NEW ATTR+.ROW*.NUM_COLS;
357 0613 DECR C FROM .NUM_COLS TO -1 DO
358 0614 BEGIN
359 0615 PTEXT=.PTEXT-1;
360 0616 PATTR=.PATTR-1;
361 0617 BEGIN
362 0618 BIND TEXT_CHAR=.PTEXT : BYTE,
363 0619 ATTR_CHAR=.PATTR : BYTE;
364 0620 IF .TEXT_CHAR EQL 'C' AND .ATTR_CHAR EQL 0
365 0621 THEN BLANK_COL=.C
366 0622 ELSE EXITLOOP
367 0623 END;
368 0624 END;
369 0625
370 0626 WHILE .CHARS_LEFT NEQ 0 DO
371 0627 BEGIN T scan
372 0628 IF .CUR_TEXT_PTR[0] EQL .NEW_TEXT_PTR[0] AND
373 0629 .CUR_ATTR_PTR[0] EQL .NEW_ATTR_PTR[0]
374 0630 THEN BEGIN ! Characters agree
375 0631 CUR_TEXT_PTR=.CUR_TEXT_PTR+1;
376 0632 CUR_ATTR_PTR=.CUR_ATTR_PTR+1;
377 0633 NEW_TEXT_PTR=.NEW_TEXT_PTR+1;
378 0634 NEW_ATTR_PTR=.NEW_ATTR_PTR+1;
379 0635 CHARS_LEFT=.CHARS_LEFT-1
380 0636 END ! Characters agree
381 0637 ELSE BEGIN ! Characters disagree
382 0638
383 0639 INDEX=.CUR_TEXT_PTR-CUR_TEXT;
384 0640
385 0641 !+
386 0642 ! Re-express address as a row and column number
387 0643 !-
388 0644
389 0645 $MAKE_ROW_COL(INDEX,ROW,COL);
390 0646
391 0647 COL=.CUR_TEXT_PTR-.PRE_PTR_IN_ROW;
392 0648
393 0649 !+
394 0650 ! At this point, the cursor is positioned at
395 0651 ! .CURSOR_ROW, .CURSOR_COL. The first character that
396 0652 ! needs to be rewritten is at .ROW, .COL.
397 0653 ! Determine a minimal update sequence to get us from
398 0654 ! where cursor is to where it needs to be to do rewrite.
399 0655 !-
400 0656
401 0657 !+
402 0658 ! Set the column to "unknown" if we are past the end of
403 0659 ! the terminal width. We cannot assume that the cursor
404 0660 ! has become stuck in the last column, because the
405 0661 ! user may have done a SET TERMINAL/WIDTH=n command
406 0662 ! to shorten his logical terminal width.
407 0663 !-
408 0664
409 0665 IF .CURSOR_COL GTRU .NUM_COLS
410 0666 THEN CURSOR_COL=0;
411 0667
412 0668 SMG$FIND_MIN_CURSOR_POS ( PBCB, ! Pasteboard Control block
```



```
413 0669 S
414 0670
415 0671 .CURSOR_ROW, | Current row
416 0672 .CURSOR_COL, | Current column
417 0673 .ROW, | Desired row
418 0674 .COL, | Desired column
419
420 + Update our record of where we are on screen after
421 we output as much of the string as is currently in
422 our buffer.
423 -
424 0675
425 0676 CURSOR_ROW = .ROW ;
426 0677 CURSOR_COL = .COL ;
427 0678
428 + Now that we are positioned at first difference,
429 figure out what needs to be written.
430 -
431 0679
432 0680
433 + If we are at or past the blank pointer, then
434 just blank the remainder of the line and exit.
435 -
436 0681
437 0682 IF .CURSOR_COL GEQU .BLANK_COL
438 0683 THEN BEGIN ! erase rest of line
439 0684 LOCAL STATUS;
440 0685 STATUS=ERASE LINE(PBCB);
441 0686 IF NOT .STATUS THEN RETURN .STATUS;
442 0687 EXITLOOP
443 0688 END; ! erase rest of line
444 0689
445 0690
446 + Note that our linear position within the buffer
447 is given by the index INDEX.
448 We now calculate the linear position of the last
449 character on this row, storing the resulting index
450 in END_ROW_INDEX.
451 -
452 0691
453 0692 END_ROW_INDEX=$L(.ROW,.NUM_COLS);
454 0693
455 0694
456 + We now must search between INDEX and END_ROW_INDEX
457 for the longest sequence (all of the same rendition)
458 of changed characters.
459 -
460 0695
461 0696
462 + Step 1: find the longest sequence of characters
463 that are all of the same rendition.
464 Put our currently desired attributes in RENDITION.
465 -
466 0697
467 0698 RENDITION = .NEW ATTR[.INDEX];
468 0699 FINAL_INDEX = .END_ROW_INDEX+1;
469 0700
```

```
470 0726 5
471 0727
472 0728
473 0729
474 0730
475 0731
476 0732
477 0733
478 0734
479 0735
480 0736
481 0737
482 0738
483 0739
484 0740
485 0741
486 0742
487 0743
488 0744
489 0745
490 0746
491 0747
492 0748
493 0749
494 0750
495 0751
496 0752
497 0753
498 0754
499 0755
500 0756
501 0757
502 0758
503 0759
504 0760
505 0761
506 0762
507 0763
508 0764
509 0765
510 0766
511 0767
512 0768
513 0769
514 0770
515 0771
516 0772
517 0773
518 0774
519 0775
520 0776
521 0777
522 0778
523 0779
524 0780
525 0781
526 0782

!+
! Set up FINAL_INDEX to be the first index past
! the longest such difference sequence.
!-

INCR I FROM .INDEX+1 TO .END_ROW_INDEX DO
    BEGIN ! scan for end of change
    IF (.NEW_TEXT[.I] EQL .CUR_TEXT[.I] AND
        .NEW_ATTR[.I] EQL .CUR_ATTR[.I])
    OR .NEW_ATTR[.I] NEQ .RENDITION
    THEN BEGIN ! end-of-change
        FINAL_INDEX=.I;
        EXITLOOP
    END; ! end-of-change
    END; ! scan for end of change

!+
! We now must update the screen from .INDEX to .FINAL_INDEX-1
! positions using the attributes stored in RENDITION.
! The final SPACE_COUNT positions are to be erased.
!-

LEN=.FINAL_INDEX-.INDEX;

IF .LEN GTRU 0
    THEN BEGIN ! output revised sequence
        $OUTPUT_STRING(.LEN,.NEW_TEXT_PTR,.RENDITION);
        CURSOR_COL=.CURSOR_COL+.LEN
    END; ! output revised sequence

!+
! Update our pointers and the number of chars left.
!-

CUR_TEXT_PTR =.CUR_TEXT_PTR+.LEN;
CUR_ATTR_PTR =.CUR_ATTR_PTR+.LEN;
NEW_TEXT_PTR =.NEW_TEXT_PTR+.LEN;
NEW_ATTR_PTR =.NEW_ATTR_PTR+.LEN;

CHARS_LEFT=.CHARS_LEFT-.LEN

END ! Characters disagree

END; ! scan
END; ! scan row .ROW

!+
! Make the two buffers agree.
! The screen now contains what we think should be there.
!-

CHSMOVE(.SIZE,NEW_TEXT,CUR_TEXT);
CHSMOVE(.SIZE,NEW_ATTR,CUR_ATTR);
CHSMOVE(.NUM_ROWS-1,NEW_LCV,CUR_LCV);

!+
! Move the cursor to the place where the user thinks it is.
```

```

0783 2 ! (But only if we are not already there.)
0784 2 !
0785 2
0786 2 IF .CUR_LCVC.NEW_CURSOR_ROW] NEQ 0
0787 2 THEN ADJUSTED_COL=.CURSOR_COL
0788 2 ELSE ADJUSTED_COL=2*.CURSOR_COL-1;
0789 2
0790 2 OLD_CURSOR_ROW=.CURSOR_ROW;
0791 2 OLD_CURSOR_COL=.CURSOR_COL;
0792 2
0793 2 SMG$UPDATE_PHYSICAL_CURSOR(PBCB);
0794 2
0795 2 RETURN SSS_NORMAL
0796 2
0797 1 END;
! End of routine SMG$OUTPUT_MINIMAL_UPDATE

```

```

.TITLE SMG$MIN Minimal update calculation
.IDENT \1-016\

.EXTRN SMG$OUTPUT, SMG$GET_TERM_DATA
.EXTRN SMG$SPUT_SCREEN

.PSECT _SMG$CODE,NOWRT, SHR, PIC.2

.ENTRY SMG$OUTPUT_MINIMAL_UPDATE, Save R2,R3,R4,- 0408
R5,R6,R7,R8,R9,R10,R11
MOVAB -320(SP), SP
MOVL P PBCB, R9 0451
MOVL 8(R9), R10 0452
PUSHL 20(R10) 0455
PUSHL 12(R10) 0458
BBC #6, 208(R9), 1$ 0505
MOVC5 #0, (SP), #0, 40(R10), @4(SP) 0507

MOVW #1, 168(R9) 0508
MOVW #1, 172(R9) 0509
MOVW 2(R10), 170(R9) 0510
MOVW 6(R10), 174(R9) 0511
BICB2 #64, 208(R9) 0512
CLRL CURSOR_ROW 0524
CLRL CURSOR_COL 0525
MOVZWL 170(R9), 52(SP) 0527
MOVZWL 168(R9), ROW
DECL ROW
BRW 28$
MOVAB -1(R2), R11 0533
MOVZWL 6(R10), 8(SP)
MULL2 8(SP), R11
ADDL3 4(SP), R11, CUR_TEXT_PTR
MOVAB @24(R10)[R11], CUR_ATTR_PTR 0534
MOVAB @8(R10)[R11], NEW_TEXT_PTR 0535
ADDL3 R11, (SP), NEW_ATTR_PTR 0536
MOVZBL @44(R10)[ROW], -R0 0538
BNEQ 3$
MOVL 8(SP), CHARS_LEFT 0540
BRB 4$

```

```

OFFC 00000
SE FECD CE 9E 00002
59 04 AC D0 00007
5A 08 A9 D0 0000B
14 AA DD 0000F
OC AA DD 00012
C9 06 E1 00015
6E 00 2C 0001B
04 BE 00021
00AB C9 01 B0 00023
00AC C9 01 B0 00028
00AA C9 02 AA B0 0002D
00AE C9 06 AA B0 00033
00D0 C9 40 BF 8A 00039
18 AE D4 0003F 1$:
10 AE D4 00042
34 AE 00AA C9 3C 00045
52 00AB C9 3C 0004B
52 D7 00050
024A 31 00052
SB FF A2 9E 00055 2$:
08 AE 06 AA 3C 00059
SB 08 AE C4 0005E
53 04 AE C1 00062
28 AE 18 BA4B 9E 00067
20 AE 08 BA4B 9E 0006D
AE 6E 5B C1 00073
50 2C BA4B 9A 00078
1C AE 07 12 0007D
08 AE D0 0007F
06 11 00084

```



1C	AE	08	AE	02	C7	00086	38:	DIVL3	#2, 8(SP), CHARS LEFT	0542
			55	FF	A3	9E	48:	MOVAB	-1(R3), PRE_PTR_TN_ROW	0544
			50	30	BA42	91		CMPB	248(R10)[ROW], R0	0550
					03	12		BNEQ	58	
					00EC	31		BRW	138	
					01	DD	58:	PUSHL	#1	0565
					52	DD		PUSHL	ROW	0568
		18			AE	DD		PUSHL	CURSOR_COL	0567
		24			AE	DD		PUSHL	CURSOR_ROW	0566
					59	DD		PUSHL	R9	0565
0000V	CF				05	FB		CALLS	#5, SMGSSFIND_MIN_CURSOR_POS	
18	AE				52	DD		MOVL	ROW, CURSOR_ROW	0575
10	AE				01	DD		MOVL	#1, CURSOR_COL	0576
					50	D4		CLRL	BUFLN	0578
			50	2C	BA42	9A		MOVZBL	244(R10)[ROW], R0	0584
			01		50	91		CMPB	R0, #1	0586
					20	12		BNEQ	68	
				00FC	C9	D5		TSTL	252(R9)	
					6E	13		BEQL	98	
					3C	AE		CLRL	INPUT_ARGS	
					3C	AE		PUSHAB	INPUT_ARGS	
					0104	C9		PUSHL	260(R9)	
					0108	C9		PUSHAB	264(R9)	
					0100	C9		PUSHAB	256(R9)	
					01CE	8F		MOVZWL	#462, 28(SP)	
1C	AE				72	11		BRB	118	
					50	91	68:	CMPB	R0, #2	0587
	02				20	12		BNEQ	78	
					00FC	C9		TSTL	252(R9)	
					49	13		BEQL	98	
					3C	AE		CLRL	INPUT_ARGS	
					3C	AE		PUSHAB	INPUT_ARGS	
					0104	C9		PUSHL	260(R9)	
					0108	C9		PUSHAB	264(R9)	
					0100	C9		PUSHAB	256(R9)	
					01CD	8F		MOVZWL	#461, 28(SP)	
1C	AE				4D	11		BRB	118	
					50	91	78:	CMPB	R0, #3	0588
	03				20	12		BNEQ	88	
					00FC	C9		TSTL	252(R9)	
					24	13		BEQL	98	
					3C	AE		CLRL	INPUT_ARGS	
					3C	AE		PUSHAB	INPUT_ARGS	
					0104	C9		PUSHL	260(R9)	
					0108	C9		PUSHAB	264(R9)	
					0100	C9		PUSHAB	256(R9)	
					01CC	8F		MOVZWL	#460, 28(SP)	
1C	AE				28	11		BRB	118	
					50	D5	88:	TSTL	R0	0589
					36	12		BNEQ	128	
					00FC	C9		TSTL	252(R9)	
					06	12		BNEQ	108	
					0108	C9	98:	CLRL	264(R9)	
					2A	11		BRB	128	
					3C	AE	108:	CLRL	INPUT_ARGS	
					3C	AE		PUSHAB	INPUT_ARGS	
					0104	C9		PUSHL	260(R9)	

			0108	C9	9F	00143	PUSHAB	264(R9)		
			0100	C9	9F	00147	PUSHAB	256(R9)		
	1C	AE	023E	8F	3C	0014B	MOVZWL	#574, 28(SP)		
			1C	AE	9F	00151	PUSHAB	28(SP)		
			00FC	C9	9F	00154	PUSHAB	252(R9)		
	00000000G	00		06	FB	00158	CALLS	#6, SMG\$GET_TERM_DATA		
		01		50	E8	0015F	BLBS	STATUS, 128		
					04	00162	RET			
		50	0108	C9	D0	00163	MOVL	264(R9), R0	0596	
			1C	13		00168	BEQL	138		
			0104	C9	DD	0016A	PUSHL	260(R9)	0599	
				50	DD	0016E	PUSHL	R0	0598	
				59	DD	00170	PUSHL	R9		
	00000000G	00		03	FB	00172	CALLS	#3, SMG\$\$OUTPUT		
	38	AE		50	D0	00179	MOVL	R0, STATUS		
		05	38	AE	E8	0017D	BLBS	STATUS, 138	0600	
		50	38	AE	D0	00181	MOVL	STATUS, R0		
					04	00185	RET			
54	08	AE		01	C1	00186	ADDL3	#1, 8(SP), BLANK_COL	0610	
50		52	08	AE	C5	0018B	MULL3	8(SP), ROW, R0	0611	
	0C	AE	08	BA40	9E	00190	MOVAB	28(R10)(R0), PTEXT		
51		6E		50	C1	00196	ADDL3	R0, (SP), PATTR	0612	
50	08	AE		01	C1	0019A	ADDL3	#1, 8(SP), C	0613	
				12	11	0019F	BRB	158		
			0C	AE	D7	001A1	DECL	PTEXT	0615	
				51	D7	001A4	DECL	PATTR	0616	
			20	0C	BE	91	001A6	CMPB	@PTEXT, #32	0620
				0A	12	001AA	BNEQ	168		
				61	95	001AC	TSTB	(PATTR)		
				06	12	001AE	BNEQ	168		
	54			50	D0	001B0	MOVL	C, BLANK_COL	0621	
	EB			50	F5	001B3	SOBGTR	C, 148	0613	
			1C	AE	D5	001B6	TSTL	CHARS_LEFT	0626	
				03	12	001B9	BNEQ	188		
				00E1	31	001BB	BRW	288		
	20	BE		63	91	001BE	CMPB	(CUR_TEXT_PTR), @NEW_TEXT_PTR	0628	
				17	12	001C2	BNEQ	198		
	24	BE	28	BE	91	001C4	CMPB	@CUR_ATTR_PTR, @NEW_ATTR_PTR	0629	
				10	12	001C9	BNEQ	198		
				53	D6	001CB	INCL	CUR_TEXT_PTR	0631	
			28	AE	D6	001CD	INCL	CUR_ATTR_PTR	0632	
			20	AE	D6	001D0	INCL	NEW_TEXT_PTR	0633	
			24	AE	D6	001D3	INCL	NEW_ATTR_PTR	0634	
			1C	AE	D7	001D6	DECL	CHARS_LEFT	0635	
				DB	11	001D9	BRB	168		
			04	AE	C3	001DB	SUBL3	4(SP), CUR_TEXT_PTR, INDEX	0639	
				55	C3	001E0	SUBL3	PRE_PTR IN_ROW, CUR_TEXT_PTR, COL	0647	
			10	AE	D1	001E5	CMPB	CURSOR_COL, 8(SP)	0665	
				03	1B	001EA	BLEQU	208		
			10	AE	D4	001EC	CLRL	CURSOR_COL	0666	
			30	AE	DD	001EF	PUSHL	COL	0672	
				52	DD	001F2	PUSHL	ROW	0671	
			18	AE	DD	001F4	PUSHL	CURSOR_COL	0670	
			24	AE	DD	001F7	PUSHL	CURSOR_ROW	0669	
				59	DD	001FA	PUSHL	R9	0668	
	0000V	CF		05	FB	001FC	CALLS	#5, SMG\$\$FIND_MIN_CURSOR_POS		
	18	AE		52	D0	00201	MOVL	ROW, CURSOR_ROW	0680	

	10	AE	30	AE	DO	00205	MOVL	COL, CURSOR_COL	0681
		54	10	AE	D1	0020A	CMPL	CURSOR_COL, -BLANK_COL	0693
				OB	1F	0020E	BLSSU	21\$	
	0000V	CF		59	DD	00210	PUSHL	R9	0696
		A1		01	FB	00212	CALLS	#1, ERASE_LINE	
				50	EB	00217	BLBS	STATUS, 17\$	0697
					04	0021A	RET		
50	08	AE		01	C3	0021B	SUBL3	#1, 8(SP), R0	0709
58		5B		50	C1	00220	ADDL3	R0, R11, END_ROW_INDEX	
		50		6E	DO	00224	MOVL	(SP), R0	0723
	2C	AE		6640	9A	00227	MOVZBL	(INDEX)[R0], RENDITION	
		57		01	A8	0022C	MOVAB	1(R8), FINAL_INDEX	0724
		50		56	DO	00230	MOVL	INDEX, I	0731
				28	11	00233	BRB	25\$	
	51			04	AE	00235	MOVL	4(SP), R1	0733
	6041			08	BA40	91	CMPB	@8(R10)[I], (I)[R1]	
					OB	12	BNEQ	23\$	
	51			6E	DO	00241	MOVL	(SP), R1	0734
	18	BA40		6041	91	00244	CMPB	(I)[R1], @24(R10)[I]	
				0C	13	0024A	BEQL	24\$	
	51			6E	DO	0024C	MOVL	(SP), R1	0735
2C	AE			00	ED	0024F	CMPZV	#0, #8, (I)[R1], RENDITION	
				05	13	00256	BEQL	25\$	
		57		50	DO	00258	MOVL	I, FINAL_INDEX	0737
				04	11	0025B	BRB	26\$	0736
		50		58	F3	0025D	AOBLEQ	END ROW INDEX, I, 22\$	0731
14	D4			56	C3	00261	SUBL3	INDEX, FINAL_INDEX, LEN	0748
	AE	57		1C	13	00266	BEQL	27\$	0750
				2C	AE	DD	PUSHL	RENDITION	0752
					7E	7C	CLRQ	-(SP)	
				2C	AE	DD	PUSHL	NEW_TEXT_PTR	
				24	AE	DD	PUSHL	LEN	
					59	DD	PUSHL	R9	
	00000000G	00		06	FB	00275	CALLS	#6, SMGSSPUT_SCREEN	
		6E		50	E9	0027C	BLBC	STATUS, 31\$	
	10	AE		14	AE	CO	ADDL2	LEN, CURSOR_COL	0753
		53		14	AE	CO	ADDL2	LEN, CUR_TEXT_PTR	0760
	28	AE		14	AE	CO	ADDL2	LEN, CUR_ATTR_PTR	0761
	20	AE		14	AE	CO	ADDL2	LEN, NEW_TEXT_PTR	0762
	24	AE		14	AE	CO	ADDL2	LEN, NEW_ATTR_PTR	0763
	1C	AE		14	AE	C2	SUBL2	LEN, CHARS_LEFT	0765
				FF17	31	0029C	BRW	16\$	0627
FDAF				34	AE	F1	ACBL	52(SP), #1, ROW, 2\$	0527
	04	52		28	AA	28	MOVCL	40(R10), @8(R10), @4(SP)	0777
	18	BE		28	AA	28	MOVCL	40(R10), @0(SP), @24(R10)	0778
		BA		02	AA	3C	MOVZWL	2(R10), R0	0779
					50	D6	INCL	R0	
	30	BA			50	28	MOVCL	R0, @44(R10), @48(R10)	
		2C		20	AA	3C	MOVZWL	32(R10), R0	0786
				30	AA	CO	ADDL2	48(R10), R0	
					60	95	TSTB	(R0)	
					06	13	BEQL	29\$	
		50		10	AE	DO	MOVL	CURSOR_COL, ADJUSTED_COL	0787
					07	11	BRB	30\$	
	50	10			01	78	ASHL	#1, CURSOR_COL, ADJUSTED_COL	0788
					50	D7	DECL	ADJUSTED_COL	
	24	AA		18	AE	B0	MOVW	CURSOR_ROW, 36(R10)	0790



SMG\$MIN  
1-016

Minimal update calculation  
SMG\$\$OUTPUT\_MINIMAL\_UPDATE - Calculate minimum

14-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMG\$MIN.B32;1

Page 16  
(6)

26 AA 10 AE B0 002DE  
59 DD 002E3  
0000V CF 01 FB 002E5  
50 01 D0 002EA  
04 002ED 318:

MOVW CURSOR\_COL, 38(R10)  
PUSHL R9  
CALLS #1, SMG\$\$UPDATE\_PHYSICAL\_CURSOR  
MOVL #1, R0  
RET

.. 0791  
.. 0793  
.. 0795  
.. 0797

; Routine Size: 750 bytes, Routine Base: \_SMG\$CODE + 0000

```
0798 1 %SBTTL 'SMG$UPDATE_PHYSICAL_CURSOR'
0799 1 GLOBAL ROUTINE SMG$UPDATE_PHYSICAL_CURSOR (P_PBCB) =
0800 1 ++
0801 1 FUNCTIONAL DESCRIPTION:
0802 1
0803 1     This routine forces the physical cursor to move to
0804 1     a new location specified in the WCB.
0805 1     It also updates any internal structures.
0806 1     The cursor is clipped to an appropriate place if it
0807 1     falls outside the physical screen.
0808 1
0809 1 CALLING SEQUENCE:
0810 1
0811 1     ret_status.wlc.v = SMG$UPDATE_PHYSICAL_CURSOR ( P_PBCB.rab.r)
0812 1
0813 1 FORMAL PARAMETERS:
0814 1
0815 1     P_PBCB.rab.r           Address of pasteboard control block
0816 1
0817 1 IMPLICIT INPUTS:
0818 1
0819 1     WCB[WCB_W_CURR_CUR_ROW] Desired new row for physical cursor
0820 1     WCB[WCB_W_CURR_CUR_COL] Desired new col for physical cursor
0821 1     WCB[WCB_W_OLD_CUR_ROW]  Physical row where cursor now is
0822 1     WCB[WCB_W_OLD_CUR_COL]  Physical col where cursor now is
0823 1
0824 1 IMPLICIT OUTPUTS:
0825 1
0826 1     WCB[WCB_W_CURR_CUR_ROW] New cursor row
0827 1     WCB[WCB_W_CURR_CUR_COL] New cursor col
0828 1     WCB[WCB_W_OLD_CUR_ROW]  New cursor row
0829 1     WCB[WCB_W_OLD_CUR_COL]  New cursor col
0830 1
0831 1 COMPLETION STATUS:
0832 1
0833 1     $$$_NORMAL           Normal successful completion
0834 1
0835 1 SIDE EFFECTS:
0836 1
0837 1     The cursor may move to a new physical location
0838 1
0839 1
0840 1
0841 1
0842 1
0843 1
0844 1
```

```

: 591      0845 2 BEGIN
: 592      0846 2
: 593      0847 2 BIND
: 594      0848 2
: 595      0849 2 PBCB          = .P PBCB          : BLOCK[.BYTE],
: 596      0850 2 WCB          = .PBCB[PBCB_A_WCB]   : BLOCK[.BYTE],
: 597      0851 2 NUM_ROWS    = WCB[WCB-W_NO_ROWS]   : WORD,
: 598      0852 2 NUM_COLS    = WCB[WCB-W_NO_COLS]   : WORD,
: 599      0853 2 NEW_LCV     = WCB[WCB-A_LINE_CHAR]  : VECTOR[.BYTE],
: 600      0854 2 CUR_LCV     = WCB[WCB-A_SCR_LINE_CHAR] : VECTOR[.BYTE],
: 601      0855 2 OLD_CURSOR_ROW = WCB[WCB-W_OLD_CUR_ROW] : SIGNED WORD,
: 602      0856 2 OLD_CURSOR_COL = WCB[WCB-W_OLD_CUR_COL] : SIGNED WORD,
: 603      0857 2 NEW_CURSOR_ROW = WCB[WCB-W_CURR_CUR_ROW] : SIGNED WORD,
: 604      0858 2 NEW_CURSOR_COL = WCB[WCB-W_CURR_CUR_COL] : SIGNED WORD;

```



```
606 0859 2 IF .OLD_CURSOR_ROW NEQ .NEW_CURSOR_ROW
607 0860 OR .OLD_CURSOR_COL NEQ .NEW_CURSOR_COL
608 0861 THEN BEGIN
609 0862
610 0863     !+
611 0864     ! If the desired location is off the screen,
612 0865     ! Clip it to the nearest edge.
613 0866     !-
614 0867
615 0868     IF .NEW_CURSOR_ROW LSS 1
616 0869     THEN .NEW_CURSOR_ROW=1;
617 0870
618 0871     IF .NEW_CURSOR_COL LSS 1
619 0872     THEN .NEW_CURSOR_COL=1;
620 0873
621 0874     IF .NEW_CURSOR_ROW GTRU .NUM_ROWS
622 0875     THEN .NEW_CURSOR_ROW=.NUM_ROWS;
623 0876
624 0877     IF .NEW_CURSOR_COL GTRU .NUM_COLS
625 0878     THEN .NEW_CURSOR_COL=.NUM_COLS;
626 0879
627 0880     !+
628 0881     ! Physically move the cursor there.
629 0882     !-
630 0883
631 0884     SMG$$FIND_MIN_CURSOR_POS(
632 0885         PBCB,                ! Pasteboard control block
633 0886         .OLD_CURSOR_ROW,     ! Current location on screen
634 0887         .OLD_CURSOR_COL,
635 0888         .NEW_CURSOR_ROW,     ! Desired location
636 0889         .NEW_CURSOR_COL);
637 0890
638 0891     END;
639 0892
640 0893     !+
641 0894     ! Make the new and the old cursor positions agree.
642 0895     !-
643 0896
644 0897     OLD_CURSOR_ROW=.NEW_CURSOR_ROW;
645 0898     OLD_CURSOR_COL=.NEW_CURSOR_COL;
646 0899
647 0900     ! Special try:
648 0901     ! If current line is special, mark the column as unknown.
649 0902
650 0903     IF .CUR_LCVR[.NEW_CURSOR_ROW] NEQ 0
651 0904     THEN .OLD_CURSOR_COL=0;
652 0905
653 0906     RETURN SSS_NORMAL
654 0907
655 0908 END;
```

003C 00000

.ENTRY SMG\$\$UPDATE\_PHYSICAL\_CURSOR, Save R2,R3,R4,-; 0799  
R5

SMG\$MIN  
1-016

Minimal update calculation  
SMG\$\$UPDATE\_PHYSICAL\_CURSOR

C 5  
16-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGMIN.B32;1

Page 20  
(9)

50	04	AC	D0	00002	MOVL	P PBCB, R0	0849
52	08	A0	D0	00006	MOVL	8(R0), R2	0850
55	30	A2	D0	0000A	MOVL	48(R2), R5	0854
53	20	A2	9E	0000E	MOVAB	32(R2), R3	0857
54	22	A2	9E	00012	MOVAB	34(R2), R4	0858
63	24	A2	B1	00016	CMPL	36(R2), (R3)	0859
		06	12	0001A	BNEQ	1\$	
64	26	A2	B1	0001C	CMPL	38(R2), (R4)	0860
		41	13	00020	BEQL	6\$	
		63	B5	00022	1\$: TSTW	(R3)	0868
		03	14	00024	BGTR	2\$	
63		01	B0	00026	MOVW	#1, (R3)	0869
		64	B5	00029	2\$: TSTW	(R4)	0871
		03	14	0002B	BGTR	3\$	
64		01	B0	0002D	MOVW	#1, (R4)	0872
51	02	A2	3C	00030	3\$: MOVZWL	2(R2), R1	0874
10		00	EC	00034	CMPL	#0, #16, (R3), R1	
		04	1B	00039	BLEQU	4\$	
63	02	A2	B0	0003B	MOVW	2(R2), (R3)	0875
51	06	A2	3C	0003F	4\$: MOVZWL	6(R2), R1	0877
10		00	EC	00043	CMPL	#0, #16, (R4), R1	
		04	1B	00048	BLEQU	5\$	
64	06	A2	B0	0004A	MOVW	6(R2), (R4)	0878
7E		64	32	0004E	5\$: CVTWL	(R4), -(SP)	0889
7E		63	32	00051	CVTWL	(R3), -(SP)	0888
7E	26	A2	32	00054	CVTWL	38(R2), -(SP)	0887
7E	24	A2	32	00058	CVTWL	36(R2), -(SP)	0886
		50	DD	0005C	PUSHL	R0	0884
0000V	CF	05	FB	0005E	CALLS	#5, SMG\$\$FIND_MIN_CURSOR_POS	
50		63	32	00063	6\$: CVTWL	(R3), R0	0897
24	A2	50	B0	00066	MOVW	R0, 36(R2)	
26	A2	64	B0	0006A	MOVW	(R4), 38(R2)	0898
		6045	95	0006E	TSTB	(R0)[R5]	0903
		03	13	00071	BEQL	7\$	
	26	A2	B4	00073	CLRW	38(R2)	0904
50		01	D0	00076	7\$: MOVL	#1, R0	0906
		04	00079	RET			0908

; Routine Size: 122 bytes, Routine Base: \_SMG\$CODE + 02EE

```
657 0909 1 %SBTTL 'SMG$SET_PHYSICAL_CURSOR'
658 0910 1 GLOBAL ROUTINE SMG$SET_PHYSICAL_CURSOR (PBID,P_ROW,P_COL) =
659 0911 1 ++
660 0912 1 FUNCTIONAL DESCRIPTION:
661 0913 1
662 0914 1 This routine moves the physical cursor on a physical
663 0915 1 screen to a particular location.
664 0916 1
665 0917 1 CALLING SEQUENCE:
666 0918 1
667 0919 1 ret_status.wlc.v = SMG$SET_PHYSICAL_CURSOR ( PBID.rl.r,P_ROW.rl.r,
668 0920 1 P_COL.rl.r)
669 0921 1
670 0922 1 FORMAL PARAMETERS:
671 0923 1
672 0924 1 PBID.rl.r Pasteboard id
673 0925 1
674 0926 1 P_ROW.rl.r The row number to move to
675 0927 1
676 0928 1 P_COL.rl.r The column number to move to
677 0929 1
678 0930 1 IMPLICIT INPUTS:
679 0931 1
680 0932 1 NONE
681 0933 1
682 0934 1 IMPLICIT OUTPUTS:
683 0935 1
684 0936 1 NONE
685 0937 1
686 0938 1 COMPLETION STATUS:
687 0939 1
688 0940 1 SMG$WRONUMARG Wrong number of arguments
689 0941 1 SMG$INVPAS_ID Invalid pasteboard id
690 0942 1 SMG$INVROW Position is not within pasteboard (off top or bottom)
691 0943 1 SMG$INVCOL Position is not within pasteboard (off left or right)
692 0944 1 SSS_NORMAL Normal successful completion
693 0945 1
694 0946 1 SIDE EFFECTS:
695 0947 1
696 0948 1 NONE
697 0949 1 --
```



SMGSMIN  
1-016

Minimal update calculation  
SMGSSET\_PHYSICAL\_CURSOR

E 5  
16-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGMIN.B32;1

Page 22  
(11)

```

699 0950 2 BEGIN
700 0951 2 BIND
701 0952 2
702 0953 2 ROW
703 0954 2 COL
704 0955 2 = .P_ROW,
705 0956 2 = .P_COL;
706 0957 2 LOCAL
707 0958 2
708 0959 2 STATUS,
709 0960 2 PBCB
710 0961 2 WCB
711 0962 2 : REF $PBCB_DECL,
712 0963 2 : REF $WCB_DECL;
713 0964 2 EXTERNAL LITERAL
714 0965 2 SMGS_INVROW,
SMGS_INVCOL;
```

```
716 0966 2 $SMG$VALIDATE_ARGCOUNT(3,3);
717 0967
718 0968 $SMG$GET_PBCB(.PBID,PBCB);
719 0969
720 0970 WCB=.PBCB[PBCB_A_WCB];
721 0971
722 0972 BEGIN
723 0973
724 0974 BIND
725 0975
726 0976 NUM_ROWS = WCB[WCB_W_NO_ROWS] : WORD,
727 0977 NUM_COLS = WCB[WCB_W_NO_COLS] : WORD,
728 0978 CUR_ROW = WCB[WCB_W_CURR_CUR_ROW] : WORD,
729 0979 CUR_COL = WCB[WCB_W_CURR_CUR_COL] : WORD;
730 0980
731 0981 IF .ROW GTRU .NUM_ROWS
732 0982 THEN RETURN SMG$INVROW;
733 0983 IF .COL GTRU .NUM_COLS
734 0984 THEN RETURN SMG$INVCOL;
735 0985
736 0986 CUR_ROW=.ROW;
737 0987 CUR_COL=.COL;
738 0988
739 0989 END;
740 0990
741 0991 !+
742 0992 ! Immediately move it there now if batching is not in effect.
743 0993 !-
744 0994
745 0995 IF .PBCB[PBCB_L_BATCH_LEVEL] EQL 0
746 0996 THEN BEGIN ! Move cursor
747 0997 STATUS=SMG$UPDATE_PHYSICAL_CURSOR(.PBCB);
748 0998 IF NOT .STATUS THEN RETURN .STATUS
749 0999 END; ! Move cursor
750 1000
751 1001 RETURN SS$_NORMAL
752 1002
753 1003 1 END;
```

```
.EXTRN SMG$INVROW, SMG$INVCOL
.EXTRN SMG$_WRONUMARG, SMG$INVPAS_ID
.EXTRN PBD_L_COUNT, PBD_A_PBCB
.EXTRN PBD_V_PB_AVAIL
```

```
0000 00000
03 6C 91 00002
08 13 00005
50 00000000G 8F D0 00007
04 04 0000E
50 04 BC D0 0000F 1$:
11 19 00013
00000000G 00 50 D1 00015
08 14 0001C
08 00000000G 00 50 E0 0001E
50 00000000G 8F D0 00026 2$:
```

```
.ENTRY SMG$SET_PHYSICAL_CURSOR, Save nothing
CMPB (AP), #3
BEQL 1$
MOVL #SMG$_WRONUMARG, R0
RET
MOVL @PBID, R0
BLSS 2$
CMLP R0, PBD_L_COUNT
BGTR 2$
BBS R0, PBD_V_PB_AVAIL, 3$
MOVL #SMG$INVPAS_ID, R0
```

```
: 0910
: 0966
```

```
: 0968
```

SMG\$MIN  
1-016

Minimal update calculation  
SMG\$SET\_PHYSICAL\_CURSOR

6 5  
16-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMG\$MIN.B32;1

Page 24  
(12)

08	BC	02	A0	51	00000000G0040	04	0002D	RET		
				50	08	A1	D0 0002E	3%:	MOVL	PBD A PBCB[R0], PBCB
				10		00	D0 00036		MOVL	8(PBCB), WCB
						08	ED 0003A		CMPZV	#0, #16, 2(WCB), @P_ROW
				50	00000000G	8F	1E 00041		BGEQU	4\$
							D0 00043		MOVL	#SMG\$_INVROW, R0
							04 0004A		RET	
0C	BC	06	A0	10		00	ED 0004B	4%:	CMPZV	#0, #16, 6(WCB), @P_COL
						08	1E 00052		BGEQU	5\$
				50	00000000G	8F	D0 00054		MOVL	#SMG\$_INVCOL, R0
							04 0005B		RET	
		20	A0		08	BC	B0 0005C	5%:	MOVW	@P_ROW, 32(WCB)
		22	A0		0C	BC	B0 00061		MOVW	@P_COL, 34(WCB)
					00A4	C1	D5 00066		TSTL	16%(PBCB)
						0A	12 0006A		BNEQ	6\$
						51	DD 0006C		PUSHL	PBCB
		FF13	CF			01	FB 0006E		CALLS	#1, SMG\$UPDATE_PHYSICAL_CURSOR
			03			50	E9 00073		BLBC	STATUS, 7\$
			50			01	D0 00076	6%:	MOVL	#1, R0
							04 00079	7%:	RET	

; Routine Size: 122 bytes, Routine Base: \_SMG\$CODE + 0368



```
755 1004 1 %SBTTL 'SMGSSFIND_MIN_CURSOR_POS - Find minimum cursor pos. sequence'
756 1005 1 GLOBAL ROUTINE SMGSSFIND_MIN_CURSOR_POS (
757 1006 1     P_PBCB,
758 1007 1     LINE_NO,
759 1008 1     COL_NO,
760 1009 1     DESIRED_LINE_NO,
761 1010 1     DESIRED_COL_NO
762 1011 1 ) =
763 1012 1
764 1013 1 **
765 1014 1 FUNCTIONAL DESCRIPTION:
766 1015 1
767 1016 1 CALLING SEQUENCE:
768 1017 1
769 1018 1     ret_status.wlc.v = SMGSSFIND_MIN_CURSOR_POS (
770 1019 1         P_PBCB.rab.r,
771 1020 1         LINE_NO.rl.v,
772 1021 1         COL_NO.rl.v,
773 1022 1         DESIRED_LINE_NO.rl.v,
774 1023 1         DESIRED_COL_NO.rl.v)
775 1024 1
776 1025 1 FORMAL PARAMETERS:
777 1026 1
778 1027 1     P_PBCB.rab.r      Address of PBCB
779 1028 1
780 1029 1     LINE_NO.rl.v      Current cursor line number
781 1030 1                     0 means it is unknown.
782 1031 1
783 1032 1     COL_NO.rl.v       Current cursor column number
784 1033 1                     0 means it is unknown.
785 1034 1
786 1035 1     DESIRED_LINE_NO.rl.v  Desired cursor line number position
787 1036 1
788 1037 1     DESIRED_COL_NO.rl.v   Desired cursor column number position
789 1038 1
790 1039 1 IMPLICIT INPUTS:
791 1040 1
792 1041 1     NONE
793 1042 1
794 1043 1 IMPLICIT OUTPUTS:
795 1044 1
796 1045 1     NONE
797 1046 1
798 1047 1 COMPLETION STATUS:
799 1048 1
800 1049 1     SS$NORMAL      Normal successful completion
801 1050 1
802 1051 1 SIDE EFFECTS:
803 1052 1
804 1053 1     NONE
805 1054 1 --
```

```

807 1055 2 BEGIN
808 1056
809 1057 BIND
810 1058
811 1059 PBCB = .P PBCB : BLOCK[.BYTE],
812 1060 WCB = .PBCB[PBCB_A_WCB] : BLOCK[.BYTE],
813 1061 NUM_ROWS = WCB[WCB_W_NO_ROWS] : WORD,
814 1062 NUM_COLS = WCB[WCB_W_NO_COLS] : WORD,
815 1063 CURR_TEXT = .WCB[WCB_A_SCR_TEXT_BUF] : VECTOR[.BYTE],
816 1064 CURR_ATTR = .WCB[WCB_A_SCR_ATTR_BUF] : VECTOR[.BYTE],
817 1065 LCV = .WCB[WCB_A_LINE_CHAR] : VECTOR[.BYTE],
818 1066 TERM_TYPE = PBCB[PBCB_B_DEVTTYPE] : BYTE;
819 1067
820 1068 LITERAL
821 1069
822 1070 INFINITY = 1000; ! Prohibitively large number, used
823 1071 ! to reject a sequence.
824 1072
825 1073 BUILTIN
826 1074
827 1075 EDIV:
828 1076
829 1077 LOCAL
830 1078
831 1079 TRIAL_STRING : VECTOR [SMG$K_LONGEST_SEQUENCE,BYTE],
832 1080 ! Buffer in which to construct string
833 1081 ! to be output.
834 1082 TS_LEN, ! Current length of TRIAL_STRING.
835 1083 ADJUSTED_WIDTH, ! Width or width/2
836 1084 SET_CUR_LEN; ! Length of the general set_cursor
837 1085 ! sequence to reposition cursor.

```

```

839 1086 2 1+
840 1087 2 1+
841 1088 2 1+
842 1089 2 1+
843 1090 2 1+
844 1091 2 1+
845 1092 2 1+
846 1093 2 1+
847 1094 2 1+
848 1095 2 1+
849 1096 2 1+
850 1097 2 1+
851 1098 2 1+
852 1099 2 1+
853 1100 2 1+
854 1101 2 1+
855 1102 2 1+
856 1103 2 1+
857 1104 2 1+
858 1105 2 1+
859 1106 2 1+
860 1107 2 1+
861 1108 2 1+
862 1109 2 1+
863 1110 2 1+
864 1111 2 1+
865 1112 2 1+
866 1113 2 1+
867 1114 2 1+
868 1115 2 1+
869 1116 2 1+
870 1117 2 1+
871 1118 2 1+
872 1119 2 1+
873 1120 2 1+
874 1121 2 1+
875 1122 2 1+
876 1123 2 1+
877 1124 2 1+
878 1125 2 1+
879 1126 2 1+
880 1127 2 1+
881 1128 2 1+
882 1129 2 1+
883 1130 2 1+
884 1131 2 1+
885 1132 2 1+
886 1133 2 1+
887 1134 2 1+
888 1135 2 1+
889 1136 2 1+
890 1137 2 1+
891 1138 2 1+
892 1139 2 1+
893 1140 2 1+
894 1141 2 1+
895 1142 2 1+

1+
If the current position is unknown,
then we must use the most general sequence.

1+
IF .LINE_NO EQL 0
OR .COL_NO EQL 0
THEN RETURN SET_CURSOR(PBCB,.DESIRED_LINE_NO,.DESIRED_COL_NO,.LINE_NO);

1+
General strategy is to come up with a sequence of characters that
will position us to the desired line and column number in less
characters than a set_cursor sequence will need.
The short-cut sequences to get to a specific line include:
1. <LF's> to move down the screen.
The short-cut sequences to get to a specific column include:
1. <TAB> to tab-stop immediately before desired column and
repeat a number of the current characters until we get to
desired column position.
2. <TAB> to tab-stop immediately beyond desired column and
follow that by a number of <BS's> to get to the desired column.
If at any point the trial sequence of characters gets to be
greater than the set_cursor sequence, abandon the effort and use the
set_cursor sequence.

TS_LEN = 0;                                ! Length of string constructed so far

1+
Calculate what the cost of a set_cursor sequence is will be for the
desired line and column number. This will give us the lower bound we
must beat if an alternate sequence is better.

1+
$SMG$GET_TERM_DATA(SET_CURSOR_APS,.DESIRED_LINE_NO,.DESIRED_COL_NO);
SET_CUR_LEN = .PBCB[PBCB_L_CAP_LENGTH];

1+
Now see if we are already on the proper line.

IF .LINE_NO NEQ .DESIRED_LINE_NO
THEN
    BEGIN ! Adjust line number
    IF .DESIRED_LINE_NO LSS .LINE_NO
    THEN
        BEGIN ! Move upward
            1+
            No choice -- must use general cursor sequencing to move
            upward. Output general set_cursor sequence
            (using DESIRED_LINE_NO and
            DESIRED_COL_NO) and return to caller.
            1+
        RETURN SET_CURSOR(PBCB,.DESIRED_LINE_NO,.DESIRED_COL_NO,.LINE_NO)
    END
END
```



```
      END          ! Move upward
ELSE
  BEGIN          ! Move downward
  LOCAL
    WIDE_WARNING, ! TRUE if spanning across a wide line
    LINES_DOWN ;  ! No. of lines down we need to move

    +
    See if we can reach DESIRED_LINE_NO in a number of <LF's>
    which is less than the number of characters in the
    set cursor sequence.
    We do not permit line feed through the bottom of the scrolling
    region, since the cursor would not be able to cross it that way
    (and it would cause a scroll to occur).
    We do not permit line feed through a double wide (or double high)
    line, because in some cases, this doesn't work. In particular,
    on a VT100, if you are in column 60, say and line feed down
    through a double wide line, when you get back to a single
    wide line, the cursor has now gotten to column 40!
    -

    LINES_DOWN = .DESIRED_LINE_NO - .LINE_NO;

    +
    Set WIDE_WARNING to TRUE if we would cross through or into or
    from a wide line. Double high lines are considered to be wide.
    -

    WIDE_WARNING=0;
    IF .LCV[0] NEQ 0
    THEN
      INCR L FROM .LINE_NO TO .DESIRED_LINE_NO DO
        IF .LCV[L] NEQ 0
        THEN BEGIN
          WIDE_WARNING=1;
          EXIT[COOP]
          END;

    IF (.LINES_DOWN LSS .SET_CUR_LEN) AND
      (.LINE_NO + .LINES_DOWN LEQU .PBCB[PBCB_W_BOT_SCROLL_LINE]
      OR .LINE_NO GTRU .PBCB[PBCB_W_BOT_SCROLL_LINE]) AND
      (NOT .WIDE_WARNING)
    THEN
      BEGIN ! Do it with <LF's>
        +
        Put (.LINES_DOWN) <LF's> into TRIAL_STRING and set
        TS_LEN to .LINES_DOWN.
        -
        CH$FILL (LF, .LINES_DOWN, TRIAL_STRING);
        TS_LEN = .LINES_DOWN;
        END ! Do it with <LF's>
      ELSE
        BEGIN ! Too far
          +
          Too far down or we would be crossing a lower scroll
          boundary or a wide line -- use general set cursor sequence
          -
```

```
953 1200 5 RETURN SET_CURSOR(PBCB,.DESIRED_LINE_NO,.DESIRED_COL_NO,.LINE_NO)
954 1201 END; ! Too far
955 1202 END; ! Move downward
956 1203 END; ! Adjust line number
957 1204
958 1205
959 1206
960 1207
961 1208
962 1209
963 1210
964 1211 IF .COL_NO NEQ .DESIRED_COL_NO
965 1212 THEN
966 1213 BEGIN ! Column adjustment
967 1214 LOCAL
968 1215 LEAST_COST, ! Least cost among considered strategies
969 1216 BEST_STRAT, ! Best update strategy which is better
970 1217 then general cursor positioning sequence.
971 1218 INDEX, ! Index into CURR_TEXT and CURR_ATTR
972 1219 DCN_QUAD : VECTOR [2, LONG], ! Desired column number
973 1220 ! as a quadword
974 1221 DELTA_COL, ! No. of columns between where we are and where
975 1222 we want to be.
976 1223 NO_TABS, ! No. of <TAB's> to get to tab-stop before
977 1224 DESIRED_COL_NO.
978 1225 NO_RETYPES, ! No. of char's that need to be retyped if we
979 1226 tab to tab-stop before
980 1227 NO_BS; ! No. of <BS's> to get from tab-stop beyond
981 1228 DESIRED_COL_NO back to DESIRED_COL_NO.
982 1229
983 1230
984 1231
985 1232
986 1233
987 1234
988 1235
989 1236
990 1237
991 1238
992 1239
993 1240
994 1241
995 1242
996 1243
997 1244
998 1245
999 1246
1000 1247
1001 1248
1002 1249
1003 1250
1004 1251
1005 1252
1006 1253
1007 1254
1008 1255
1009 1256
```

Reach here when we have constructed the minimal sequence to reach the desired line --not using general cursor addressing sequence. TS\_LEN tells us how long that sequence is.

Construct short-cut sequence to position to desired column number.

If earlier on line, 3 strategies are possible:

1. Do it with backspaces
2. Do it with <CR> and <TAB's> to tab-stop before followed by retypes.
3. Do it with <CR> and <TAB's> to tab-stop beyond followed by <BS's>.

If later on line, 3 strategies are possible:

4. Do it with retypes.
5. Do it with <TAB's> to tab-stop before followed by retypes.
6. Do it with <TAB's> to tab-stop after followed by <BS's>.

Calc. no of <TAB's> needed to get to tab-stop before DESIRED\_COL\_NO and the no. of subsequent retypes needed.

DCN\_QUAD [0] = .DESIRED\_COL\_NO -1;  
DCN\_QUAD [1] = 0;  
EDIV ( %REF(8), DCN\_QUAD[0], NO\_TABS, NO\_RETYPES);

If terminal doesn't support tabs,

```
1010 1257 | or user doesn't want them,
1011 1258 | then set NO_TABS to infinity.
1012 1259 |
1013 1260 |
1014 1261 IF .PBCB[PBCB_V_NOTABS] OR NOT .PBCB[PBCB_V_TABS]
1015 1262 THEN NO_TABS=INFINITY;
1016 1263 |
1017 1264 |
1018 1265 | +
1019 1266 | Calc. number of <BS's> needed if we go to tab-stop after
1020 1267 | DESIRED_COL_NO. This strategy can't be followed if the
1021 1268 | next tab stop is off past the right of the screen. In
1022 1269 | that case, we make NO_BS prohibitively large.
1023 1270 |
1024 1271 |
1025 1272 IF .LCVE[DESIRED_LINE_NO] NEQ 0
1026 1273 THEN ADJUSTED_WIDTH=.NUM_COLS/2
1027 1274 ELSE ADJUSTED_WIDTH=.NUM_COLS;
1028 1275 |
1029 1276 IF (.NO_TABS+1)*8+1 LSSU .ADJUSTED_WIDTH
1030 1277 THEN NO_BS = 8 - .NO_RETYPES
1031 1278 ELSE NO_BS = INFINITY;
1032 1279 |
1033 1280 | +
1034 1281 | Set NO_BS to infinity if the terminal does not support backspacing.
1035 1282 |
1036 1283 |
1037 1284 IF NOT .PBCB[PBCB_V_BS]
1038 1285 THEN NO_BS=INFINITY;
1039 1286 |
1040 1287 | +
1041 1288 | In case we need to do retypes, calc. where in CURR_TEXT and
1042 1289 | CURR_ATTR we need to look.
1043 1290 |
1044 1291 INDEX = $L ( .DESIRED_LINE_NO, ((.NO_TABS*8) + 1));
1045 1292 |
1046 1293 IF .DESIRED_COL_NO LEQ .COL_NO
1047 1294 THEN
1048 1295 BEGIN ! Earlier in line
1049 1296 LOCAL
1050 1297 |
1051 1298 S1_COST, S2_COST, S3_COST; | Cost of strategies
1052 1299 | S1: just BS
1053 1300 | S2: tabs then retype
1054 1301 | S3: tabs then BS
1055 1302 |
1056 1303 ! Find the cost of strategies for moving back in line
1057 1304 |
1058 1305 IF .PBCB[PBCB_V_BS]
1059 1306 THEN
1060 1307 S1_COST = .COL_NO - .DESIRED_COL_NO ! No of <BS's>
1061 1308 ELSE
1062 1309 S1_COST=INFINITY;
1063 1310 |
1064 1311 S2_COST = 1 | For <CR>
1065 1312 + .NO_TABS | For no. of tabs to tab-stop
1066 1313 | before
```



```
1067 1314 4      + .NO_RETYPES;      ! For no. of retypes
1068 1315 4
1069 1316 4      S3_COST = 1      ! For <CR>
1070 1317 4      + .NO_TABS + 1  ! For no. of tabs to tab-stop
1071 1318 4      ! after
1072 1319 4      + .NO_BS;      ! For no. of <BS's>
1073 1320 4
1074 1321 4      ! Find best strategy for moving backward in line
1075 1322 4
1076 1323 4      BEST_STRAT = 1;      LEAST_COST = .S1_COST;
1077 1324 4
1078 1325 4      IF .S2_COST LSS .LEAST_COST THEN
1079 1326 4          BEGIN BEST_STRAT = 2; LEAST_COST = .S2_COST; END;
1080 1327 4
1081 1328 4      IF .S3_COST LSS .LEAST_COST THEN
1082 1329 4          BEGIN BEST_STRAT = 3; LEAST_COST = .S3_COST; END;
1083 1330 4      END ! Earlier in line
1084 1331 4
1085 1332 4      ELSE
1086 1333 4
1087 1334 4          BEGIN      ! Later in line
1088 1335 4              LOCAL
1089 1336 4                  S4_COST, S5_COST, S6_COST;      ! Cost of strategies
1090 1337 4
1091 1338 4              ! Find costs of strategies for moving forward in line
1092 1339 4
1093 1340 4              S4_COST = .DESIRED_COL_NO - .COL_NO; ! For just retypes
1094 1341 4
1095 1342 4              IF (.NO_TABS * 8) + 1 GTR .COL_NO AND .PBCB[PBCB_V_TABS]
1096 1343 4                  AND NOT .PBCB[PBCB_V_NOTABS]
1097 1344 4              THEN
1098 1345 4                  BEGIN ! Tabbing forward is possible
1099 1346 4                      LOCAL
1100 1347 4                          COL_QUAD : VECTOR [2, LONG], ! COL_NO as quadword
1101 1348 4                          NEW_NO_TABS,
1102 1349 4                          NEW_NO_RETYPES;
1103 1350 4
1104 1351 4                          COL_QUAD [0] = .COL_NO - 1;
1105 1352 4                          COL_QUAD [1] = 0;
1106 1353 4                          EDIV (XREF(8), COL_QUAD [0], NEW_NO_TABS, NEW_NO_RETYPES);
1107 1354 4                          NO_TABS = .NO_TABS - .NEW_NO_TABS;
1108 1355 4                          S5_COST = .NO_TABS      ! For no. of tabs to tab-stop
1109 1356 4                              ! before from current position
1110 1357 4                              + .NO_RETYPES; ! for no. of retypes
1111 1358 4
1112 1359 4                          S6_COST = .NO_TABS + 1 ! For no. of tabs to tab-stop
1113 1360 4                              ! after from current position
1114 1361 4                              + .NO_BS;      ! for no. of <BS's>
1115 1362 4                      END      ! Tabbing forward is possible
1116 1363 4              ELSE
1117 1364 4                  BEGIN ! Tabbing forward not possible
1118 1365 4                      S5_COST = INFINITY;      ! Set to prohibitive value
1119 1366 4                      S6_COST = INFINITY;      ! Set to prohibitive value
1120 1367 4                  END;      ! Tabbing forward not possible
1121 1368 4
1122 1369 4              ! Find best strategy
1123 1370 4
```

```
1124 1371 4      BEST_STRAT = 4;          LEAST_COST = .S4_COST;
1125 1372 4
1126 1373 4      IF .S5_COST LSS .LEAST_COST THEN
1127 1374 4          BEGIN BEST_STRAT = 5; LEAST_COST = .S5_COST; END;
1128 1375 4
1129 1376 4      IF .S6_COST LSS .LEAST_COST THEN
1130 1377 4          BEGIN BEST_STRAT = 6; LEAST_COST = .S6_COST; END;
1131 1378 3      END;          ! Later in line
1132 1379 3
1133 1380 3      IF .TS_LEN + .LEAST_COST GTR .SET_CUR_LEN
1134 1381 3      THEN
1135 1382 4          BEGIN          ! Abandon effort
1136 1383 4          RETURN SET_CURSOR(PBCB,.DESIRED_LINE_NO,.DESIRED_COL_NO,.LINE_NO)
1137 1384 3          END;          ! Abandon effort
1138 1385 3
1139 1386 3      CASE .BEST_STRAT FROM 1 TO 6 OF
1140 1387 3      SET
1141 1388 4          [1]:BEGIN          ! Backspaces only.
1142 1389 4              NO_BS = .COL_NO - .DESIRED_COL_NO;
1143 1390 4              CH$FILL ( BS, .NO_BS, TRIAL_STRING [.TS_LEN]);
1144 1391 4              TS_LEN = .TS_LEN + .NO_BS;
1145 1392 3              END;          ! Backspace only.
1146 1393 3
1147 1394 4          [2]:BEGIN          ! <CR>, <TAB's> to tab-stop before, retypes.
1148 1395 4
1149 1396 4              !+
1150 1397 4              ! If there are actually characters to be retyped and
1151 1398 4              ! attributes are involved, give up and resort to general
1152 1399 4              ! cursor positioning sequence.
1153 1400 4              ! It will cost us too much to select-graphic-rendition
1154 1401 4              ! and undo select graphic rendition.
1155 1402 4              !-
1156 1403 4
1157 1404 4              IF .NO_RETYPES NEQ 0 AND
1158 1405 4                  CH$COMPARE (0, 0, ! len, addr
1159 1406 4                      .NO_RETYPES, CURR_ATT[.INDEX],
1160 1407 4                      0 ! fill
1161 1408 4                      ) NEQ 0
1162 1409 4              THEN
1163 1410 4                  RETURN SET_CURSOR(PBCB,.DESIRED_LINE_NO,.DESIRED_COL_NO,.LINE_NO);
1164 1411 4
1165 1412 4              TRIAL_STRING [.TS_LEN] = CR;
1166 1413 4              TS_LEN = .TS_LEN + 1;
1167 1414 4              CH$FILL ( TAB, .NO_TABS, TRIAL_STRING [.TS_LEN]);
1168 1415 4              TS_LEN = .TS_LEN + .NO_TABS;
1169 1416 4              CH$MOVE ( .NO_RETYPES, CURR_TEXT [.INDEX],
1170 1417 4                  TRIAL_STRING [.TS_LEN]);
1171 1418 4              TS_LEN = .TS_LEN + .NO_RETYPES;
1172 1419 3              END;          ! <CR>, <TAB's> to tab-stop before, retypes.
1173 1420 3
1174 1421 4          [3]:BEGIN          ! <CR>, <TAB's> to tab-stop after, <BS's>
1175 1422 4              TRIAL_STRING [.TS_LEN] = CR;
1176 1423 4              TS_LEN = .TS_LEN + 1;
1177 1424 4              CH$FILL ( TAB, .NO_TABS + 1, TRIAL_STRING [.TS_LEN]);
1178 1425 4              TS_LEN = .TS_LEN + .NO_TABS + 1;
1179 1426 4              CH$FILL ( BS, .NO_BS, TRIAL_STRING [.TS_LEN]);
1180 1427 4              TS_LEN = .TS_LEN + .NO_BS;
```

1181 1428 3  
1182 1429 3  
1183 1430 4  
1184 1431 4  
1185 1432 4  
1186 1433 4  
1187 1434 4  
1188 1435 4  
1189 1436 4  
1190 1437 4  
1191 1438 4  
1192 1439 4  
1193 1440 4  
1194 1441 4  
1195 1442 4  
1196 1443 4  
1197 1444 4  
1198 1445 4  
1199 1446 4  
1200 1447 4  
1201 1448 4  
1202 1449 4  
1203 1450 4  
1204 1451 4  
1205 1452 4  
1206 1453 3  
1207 1454 3  
1208 1455 4  
1209 1456 4  
1210 1457 4  
1211 1458 4  
1212 1459 4  
1213 1460 4  
1214 1461 4  
1215 1462 4  
1216 1463 4  
1217 1464 4  
1218 1465 4  
1219 1466 4  
1220 1467 4  
1221 1468 4  
1222 1469 4  
1223 1470 4  
1224 1471 4  
1225 1472 4  
1226 1473 4  
1227 1474 4  
1228 1475 4  
1229 1476 4  
1230 1477 4  
1231 1478 3  
1232 1479 3  
1233 1480 4  
1234 1481 4  
1235 1482 4  
1236 1483 4  
1237 1484 4

```
END;      ! <CR>, <TAB's> to tab-stop after, <BS's>

[4]:BEGIN  ! Retypes only.

      +
      | If there are actually characters to be retyped and
      | attributes are involved, give up and resort to general
      | cursor positioning sequence.
      | It will cost us too much to select-graphic-rendition
      | and undo select graphic rendition.
      -

      NO RETYPES = .DESIRED_COL_NO - .COL_NO;
      INDEX = $L ( .DESIRED_LINE_NO, .COL_NO);
      IF .NO RETYPES NEQ 0 AND
          CH$COMPARE (0, 0, ! len, addr
                      .NO RETYPES, CURR_ATTR[.INDEX],
                      0 ! fill
                      ) NEQ 0
      THEN
          RETURN SET_CURSOR(PBCB,.DESIRED_LINE_NO,.DESIRED_COL_NO,.LINE_NO);

      CH$MOVE ( .NO RETYPES, CURR_TEXT [.INDEX],
                TRIAL_STRING [.TS_LEN]);
      TS_LEN = .TS_LEN + .NO RETYPES;
      END;      ! Retypes only.

[5]:BEGIN  ! <TAB's> to tab-stop before, retypes.

      +
      | If there are actually characters to be retyped and
      | attributes are involved, give up and resort to general
      | cursor positioning sequence.
      | It will cost us too much to select-graphic-rendition
      | and undo select graphic rendition.
      -

      IF .NO RETYPES NEQ 0 AND
          CH$COMPARE (0, 0, ! len, addr
                      .NO RETYPES, CURR_ATTR[.INDEX],
                      0 ! fill
                      ) NEQ 0
      THEN
          RETURN SET_CURSOR(PBCB,.DESIRED_LINE_NO,.DESIRED_COL_NO,.LINE_NO);

      CH$FILL ( TAB, .NO TABS, TRIAL_STRING [.TS_LEN]);
      TS_LEN = .TS_LEN + .NO TABS;
      CH$MOVE ( .NO RETYPES, CURR_TEXT [.INDEX],
                TRIAL_STRING [.TS_LEN]);
      TS_LEN = .TS_LEN + .NO RETYPES;
      END;      ! <TAB's> to tab-stop before, retypes.

[6]:BEGIN  ! <TAB's> to tab-stop after, <BS's>.
      CH$FILL ( TAB, .NO TABS + 1, TRIAL_STRING [.TS_LEN]);
      TS_LEN = .TS_LEN + .NO TABS + 1;
      CH$FILL ( BS, .NO BS, TRIAL_STRING [.TS_LEN]);
      TS_LEN = .TS_LEN + .NO BS;
```

```

1238 1485      END;      ! <TAB's> to tab-stop after, <BS's>.
1239 1486      TES;
1240 1487      END;      ! Column adjustment
1241 1488
1242 1489
1243 1490      +
1244 1491      At this point in the code we have a proper sequence of characters to
1245 1492      reposition the cursor from .LINE_NO/.COL_NO to .DESIRED_LINE_NO/
1246 1493      .DESIRED_COL_NO with a relatively minimum number of characters.
1247 1494      This sequence of characters is contained in TRIAL_STRING and its
1248 1495      length is contained in .TS_LEN
1249 1496      We output this string to the screen.
1250 1497      -
1251 1498      $OUTPUT_STRING(.TS_LEN,TRIAL_STRING,0);
1252 1499
1253 1500      RETURN SS$_NORMAL
1254 1501
1255 1502      END;      ! End of routine SMG$FIND_MIN_CURSOR_POS

```

			OFFC 00000	.ENTRY	SMG\$FIND_MIN_CURSOR_POS. Save R2,R3,R4,R5,-	
					R6,R7,R8,R9,R10,R11	1005
		5E	FEE8	CE 9E 00002	MOVAB	-280(SP), SP
			04	AC DD 00007	PUSHL	P_PBCB
50		6E		08 C1 0000A	ADDL3	#8, (SP), R0
		5B		60 D0 0000E	MOVL	(R0), R1
			08	AC D5 00011	TSTL	LINE_NO
			0C	7D 13 00014	BEQL	3\$
				AC D5 00016	TSTL	COL_NO
				78 13 00019	BEQL	3\$
				57 D4 0001B	CLRL	TS_LEN
50		6E	00000FC	8F C1 0001D	ADDL3	#252, (SP), R0
				60 D5 00025	TSTL	(R0)
				0C 12 00027	BNEQ	1\$
52		6E	00000108	8F C1 00029	ADDL3	#264, (SP), R2
				62 D4 00031	CLRL	(R2)
				4C 11 00033	BRB	2\$
	10	AE		02 D0 00035	MOVL	#2, INPUT_ARGS
	14	AE	10	AC 7D 00039	MOVQ	DESIRED_LINE_NO, INPUT_ARGS+4
			10	AE 9F 0003E	PUSHAB	INPUT_ARGS
53	04	AE	00000104	8F C1 00041	ADDL3	#260, -4(SP), R3
				63 DD 0004A	PUSHL	(R3)
52	08	AE	00000108	8F C1 0004C	ADDL3	#264, 8(SP), R2
				52 DD 00055	PUSHL	R2
50	0C	AE	00000100	8F C1 00057	ADDL3	#256, 12(SP), R0
				50 DD 00060	PUSHL	R0
	18	AE	023A	8F 3C 00062	MOVZWL	#570, 24(SP)
			18	AE 9F 00068	PUSHAB	24(SP)
50	14	AE	000000FC	8F C1 0006B	ADDL3	#252, 20(SP), R0
				50 DD 00074	PUSHL	R0
	00000000G	00		06 FB 00076	CALLS	#6, SMG\$GET_TERM_DATA
		01		50 E8 0007D	BLBS	STATUS, 2\$
				04 00080	RET	
	08	AE		62 D0 00081	MOVL	(R2), SET_CUR_LEN



	10	AC	08	AC	D1	00085	CMPL	LINE_NO, DESIRED_LINE_NO	1127
				65	13	0008A	BEQL	9\$	
	08	AC	10	AC	D1	0008C	CMPL	DESIRED_LINE_NO, LINE_NO	1130
				03	18	00091	BGEQ	4\$	
				023E	31	00093	BRW	39\$	
56	10	AC	08	AC	C3	00096	SUBL3	LINE_NO, DESIRED_LINE_NO, LINES_DOWN	1164
				51	D4	0009C	CLRL	WIDE_WARNING	1171
			2C	BB	95	0009E	TSTB	044(R11)	1172
				17	13	000A1	BEQL	7\$	
50	08	AC		01	C3	000A3	SUBL3	#1, LINE_NO, L	1174
				08	11	000A8	BRB	6\$	
			2C	BB40	95	000AA	TSTB	044(R11)[L]	1175
				05	13	000AE	BEQL	6\$	
		51		01	D0	000B0	MOVL	#1, WIDE_WARNING	1177
				05	11	000B3	BRB	7\$	1176
FO		50	10	AC	F3	000B5	AOBLEQ	DESIRED LINE NO, L, 5\$	1175
	08	AE		56	D1	000BA	CMPL	LINES_DOWN, SET_CUR_LEN	1181
				D3	18	000BE	BGEQ	3\$	
50		56	08	AC	C1	000C0	ADDL3	LINE_NO, LINES_DOWN, R0	1182
52		6E	000000F6	8F	C1	000C5	ADDL3	#246, (SP), R2	
62		10		00	ED	000CD	CMPZV	#0, #16, (R2), R0	
				10	1E	000D2	BGEQU	8\$	
50		6E	000000F6	8F	C1	000D4	ADDL3	#246, (SP), R0	1183
08	AC	60		00	ED	000DC	CMPZV	#0, #16, (R0), LINE_NO	
				AF	1E	000E2	BGEQU	3\$	
				51	E8	000E4	BLBS	WIDE WARNING, 3\$	1184
56		0A		00	2C	000E7	MOVCS	#0, (SP), #10, LINES_DOWN, TRIAL_STRING	1191
				AE		000EC			
		57	1C	56	D0	000EE	MOVL	LINES DOWN, TS_LEN	1192
		54	0C	AC	D0	000F1	MOVL	COL_NO, R4	1211
	14	AC		54	D1	000F5	CMPL	R4, DESIRED_COL_NO	
				03	12	000F9	BNEQ	10\$	
				0218	31	000FB	BRW	46\$	
				01	C3	000FE	SUBL3	#1, DESIRED_COL_NO, DCN_QUAD	1251
14	AE	14	AC	AE	D4	00104	CLRL	DCN_QUAD+4	1252
				08	7B	00107	EDIV	#8, DCN_QUAD, NO_TABS, NO_RETYPES	1253
59		56	14	AE	0C	0010D	ADDL3	#12, (SP), R0	1261
		50		6E	C1	00110	BBS	#3, (R0), 11\$	
		0C		60	03	00111	ADDL3	#250, (SP), R1	
		51		6E	8F	00115	BBS	#2, (R1), 12\$	
		05		61	E0	0011D	MOVZWL	#1000, NO TABS	1262
				56	8F	00121	MOVL	DESIRED_LINE_NO, R0	1271
				50	D0	00126	TSTB	044(R11)[R0]	
					95	0012A	BEQL	13\$	
					0A	0012E	MOVZWL	6(R11), R1	1272
		51	06	AB	3C	00130	DIVL3	#2, R1, ADJUSTED_WIDTH	
52		51		02	C7	00134	BRB	14\$	
				07	11	00138	MOVZWL	6(R11), R1	1273
		51	06	AB	3C	0013A	MOVL	R1, ADJUSTED_WIDTH	
		52		51	D0	0013E	ASHL	#3, NO_TABS, R0	1275
50		56		03	78	00141	ADDL2	#9, R0	
		50		09	C0	00145	CMPL	R0, ADJUSTED_WIDTH	
		52		50	D1	00148	E 7U	15\$	
				06	1E	0014B	SUBL3	NO RETYPES, #8, NO_BS	1276
58		08		59	C3	0014D	BRB	16\$	
				05	11	00151	MOVZWL	#1000, NO BS	1277
		58	03E8	8F	3C	00153	ADDL3	#209, (SP), R0	1283
50		6E	000000D1	8F	C1	00158			

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

[illegible]

		57	08	BE	002E9							
				56	C0	002EB	41:	ADDL2	NO TABS, TS_LEN			1474
	1C AE47	14 BB4A		59	28	002EE		MOVCS	NO RETYPES, @20(R11)[INDEX], TRIAL_STRING-			1476
									[TS_LEN]			
		57		59	C0	002F6	42:	ADDL2	NO RETYPES, TS_LEN			1477
				1B	11	002F9		BRB	46:			1386
50		50	01	A6	9E	002FB	43:	MOVB	1(R6), R0			1481
	09	6E		00	2C	002FF		MOVCS	#0, (SP), #9, R0, @8(SP)			
			08	BE		00304						
		57	01	A647	9E	00306	44:	MOVB	1(NO TABS)[TS_LEN], TS_LEN			1482
58		6E		00	2C	0030B		MOVCS	#0, TSP), #8, NO_BS, TRIAL_STRING[TS_LEN]			1483
			1C	AE47		00310						
		57		58	C0	00313	45:	ADDL2	NO BS, TS_LEN			1484
				7E	7C	00316	46:	CLRQ	-(SP)			1498
				7E	D4	00318		CLRL	-(SP)			
			28	AE	9F	0031A		PUSHAB	TRIAL STRING			
				57	DD	0031D		PUSHL	TS_LEN			
			14	AE	DD	0031F		PUSHL	20(TSP)			
	00000000G	00		06	FB	00322		CALLS	#6, SMGS\$PUT_SCREEN			
		03		50	E9	00329		BLBC	STATUS, 47:			
		50		01	D0	0032C		MOVL	#1, R0			1500
					04	0032F	47:	RET				1502

; Routine Size: 816 bytes, Routine Base: \_SMGSCODE + 03E2



```
1257 1503 1 XSBTTL 'SET_CURSOR - Generate set-cursor sequence'
1258 1504 1 ROUTINE SET_CURSOR (
1259 1505 1     P_PBCB,
1260 1506 1     DESIRED_LINE_NO,
1261 1507 1     DESIRED_COL_NO,
1262 1508 1     CURRENT_ROW
1263 1509 1 ) =
1264 1510 1
1265 1511 1 ++
1266 1512 1 FUNCTIONAL DESCRIPTION:
1267 1513 1 Routine SET_CURSOR constructs the general set cursor
1268 1514 1 sequence to position to .DESIRED_LINE_NO/.DESIRED_COL_NO and outputs
1269 1515 1 it to the screen.
1270 1516 1
1271 1517 1 CALLING SEQUENCE:
1272 1518 1
1273 1519 1     ret_status.wlc.v = SET_CURSOR ( P_PBCB.rab.r,
1274 1520 1     DESIRED_LINE_NO.rl.v,
1275 1521 1     DESIRED_COL_NO.rl.v,
1276 1522 1     CURRENT_ROW.rl.v)
1277 1523 1
1278 1524 1 FORMAL PARAMETERS:
1279 1525 1
1280 1526 1     P_PBCB.rab.r      Address of PBCB
1281 1527 1
1282 1528 1     DESIRED_LINE_NO.rl.v  Desired cursor line number position
1283 1529 1
1284 1530 1     DESIRED_COL_NO.rl.v   Desired cursor column number position
1285 1531 1
1286 1532 1     CURRENT_ROW.rl.v      Current row (0 means unknown)
1287 1533 1     This matters if we are on a wide row.
1288 1534 1
1289 1535 1 IMPLICIT INPUTS:
1290 1536 1
1291 1537 1     NONE
1292 1538 1
1293 1539 1 IMPLICIT OUTPUTS:
1294 1540 1
1295 1541 1     NONE
1296 1542 1
1297 1543 1 COMPLETION STATUS:
1298 1544 1
1299 1545 1     $$$_NORMAL      Normal successful completion
1300 1546 1     errors from SMG$$OUTPUT
1301 1547 1
1302 1548 1 SIDE EFFECTS:
1303 1549 1
1304 1550 1     NONE
1305 1551 1 --
```

```

1307 1552 2 BEGIN
1308 1553
1309 1554 BIND
1310 1555
1311 1556 PBCB = .P.PBCB : BLOCK[.BYTE],
1312 1557 WCB = .PBCB[PBCB_A_WCB] : BLOCK[.BYTE],
1313 1558 LCV = .WCB[WCB_A_SCR_LINE_CHAR] : VECTOR[.BYTE];
1314 1559
1315 1560 LOCAL
1316 1561
1317 1562 STATUS; ! local status

```

```

1319 1563 1+
1320 1564 1- If we are currently on a double wide or high row (or if the
1321 1565 possibility exists) then because of bugs in the VT100 hardware,
1322 1566 we first position to column 1 of the desired line.
1323 1567 1-
1324 1568
1325 1569 IF (.CURRENT_ROW EQL 0 AND .LCV[0] NEQ 0)
1326 1570 OR .LCV[.CURRENT_ROW] NEQ 0
1327 1571 THEN BEGIN ! Move to beginning of desired line
1328 1572
1329 1573 $SMG$GET_TERM_DATA(SET_CURSOR_ABS,.DESIRED_LINE_NO,1);
1330 1574
1331 1575 1+
1332 1576 1- Output the escape sequence.
1333 1577 1-
1334 1578
1335 1579 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
1336 1580 THEN BEGIN
1337 1581 STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
1338 1582 .PBCB[PBCB_A_CAP_BUFFER]);
1339 1583 IF NOT .STATUS THEN RETURN .STATUS
1340 1584 END;
1341 1585
1342 1586 END; ! Move to beginning of desired line
1343 1587
1344 1588 1+
1345 1589 1- Create the appropriate escape sequence.
1346 1590 1-
1347 1591
1348 1592 $SMG$GET_TERM_DATA(SET_CURSOR_ABS,.DESIRED_LINE_NO,.DESIRED_COL_NO);
1349 1593
1350 1594 1+
1351 1595 1- Output the escape sequence.
1352 1596 1-
1353 1597
1354 1598 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
1355 1599 THEN BEGIN
1356 1600 STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
1357 1601 .PBCB[PBCB_A_CAP_BUFFER]);
1358 1602 IF NOT .STATUS THEN RETURN .STATUS
1359 1603 END;
1360 1604
1361 1605 RETURN SS$ _NORMAL
1362 1606
1363 1607 1- END; ! Routine SET_CURSOR

```

007C 0000 SET\_CURSOR:

56	00000000G	00	9E	00002	WORD	Save R2,R3,R4,R5,R6	1504
55	00000000G	00	9E	00009	MOVAB	SMG\$GET_TERM_DATA, R6	
5E		10	C2	00010	MOVAB	SMG\$OUTPUT, R5	
52	04	AC	D0	00013	SUBL2	#16, SP	
50	08	A2	D0	00017	MOVL	P PBCB, R2	1556
					MOVL	BTR2), R0	1557

			10	AC	D5	0001B	TSTL	CURRENT_ROW	1569
			05	12	0001E	BNEQ	1\$		
			30	B0	95	00020	TSTB	248(R0)	
			0A	12	00023	BNEQ	2\$		
50	30	A0	10	AC	C1	00025	ADDL3	CURRENT_ROW, 48(R0), R0	1570
			60	95	0002B	TSTB	(R0)		
			56	13	0002D	BEQL	5\$		
		00FC	C2	D5	0002F	TSTL	252(R2)		1573
			09	12	00033	BNEQ	3\$		
		53	0108	C2	9E	00035	MOVAB	264(R2), R3	
			63	D4	0003A	CLRL	(R3)		
			32	11	0003C	BRB	4\$		
04	AE		02	D0	0003E	MOVL	#2, INPUT_ARGS		
08	AE	08	AC	D0	00042	MOVL	DESIRED_LINE_NO, INPUT_ARGS+4		
0C	AE		01	D0	00047	MOVL	#1, INPUT_ARGS+8		
		04	AE	9F	0004B	PUSHAB	INPUT_ARGS		
		0104	C2	DD	0004E	PUSHL	260(R2)		
		53	0108	C2	9E	00052	MOVAB	264(R2), R3	
			53	DD	00057	PUSHL	R3		
		0100	C2	9F	00059	PUSHAB	256(R2)		
10	AE	023A	8F	3C	0005D	MOVZWL	#570, 16(SP)		
		10	AE	9F	00063	PUSHAB	16(SP)		
		00FC	C2	9F	00066	PUSHAB	252(R2)		
	66		06	FB	0006A	CALLS	#6, SMG\$GET_TERM_DATA		
	6E		50	E9	0006D	BLBC	STATUS, 10\$		
			63	D5	00070	TSTL	(R3)		1579
			11	13	00072	BEQL	5\$		
		0104	C2	DD	00074	PUSHL	260(R2)		1582
			63	DD	00078	PUSHL	(R3)		1581
			52	DD	0007A	PUSHL	R2		
	65		03	FB	0007C	CALLS	#3, SMG\$\$OUTPUT		
	54		50	D0	0007F	MOVL	R0, STATUS		
	52		54	E9	00082	BLBC	STATUS, 8\$		1583
		00FC	C2	D5	00085	TSTL	252(R2)		1592
			09	12	00089	BNEQ	6\$		
		53	0108	C2	9E	0008B	MOVAB	264(R2), R3	
			63	D4	00090	CLRL	(R3)		
			2E	11	00092	BRB	7\$		
04	AE		02	D0	00094	MOVL	#2, INPUT_ARGS		
08	AE	08	AC	7D	00098	MOVQ	DESIRED_LINE_NO, INPUT_ARGS+4		
		04	AE	9F	0009D	PUSHAB	INPUT_ARGS		
		0104	C2	DD	000A0	PUSHL	260(R2)		
		53	0108	C2	9E	000A4	MOVAB	264(R2), R3	
			53	DD	000A9	PUSHL	R3		
		0100	C2	9F	000AB	PUSHAB	256(R2)		
10	AE	023A	8F	3C	000AF	MOVZWL	#570, 16(SP)		
		10	AE	9F	000B5	PUSHAB	16(SP)		
		00FC	C2	9F	000B8	PUSHAB	252(R2)		
	66		06	FB	000BC	CALLS	#6, SMG\$GET_TERM_DATA		
	1C		50	E9	000BF	BLBC	STATUS, 10\$		
			63	D5	000C2	TSTL	(R3)		1598
			15	13	000C4	BEQL	9\$		
		0104	C2	DD	000C6	PUSHL	260(R2)		1601
			63	DD	000CA	PUSHL	(R3)		1600
			52	DD	000CC	PUSHL	R2		
	65		03	FB	000CE	CALLS	#3, SMG\$\$OUTPUT		
	54		50	D0	000D1	MOVL	R0, STATUS		



```
Minimal update calculation
SET_CURSOR - Generate set-cursor sequence
```

16-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 B11sg-32 V4.0-742  
[SMGRTL.SRC]SMGMIN.B32;1

Page 43  
(18)

04	54	E8	000D4		BLBS	STATUS, 9\$
50	54	D0	000D7	8\$:	MOVL	STATUS, R0
		04	000DA		RET	
50	01	D0	000DB	9\$:	MOVL	#1, R0
		04	000DE	10\$:	RET	

1602  
1605  
1607

; Routine Size: 223 bytes, Routine Base: \_SMG\$CODE + 0712

```

1365 1608 1 %SBTTL 'ERASE_LINE - Erase to end-of-line'
1366 1609 1 ROUTINE ERASE_LINE(P_PBCB) =
1367 1610 1
1368 1611 1 ++
1369 1612 1 FUNCTIONAL DESCRIPTION:
1370 1613 1
1371 1614 1 Outputs an erase-to-end-of-line sequence to the screen.
1372 1615 1
1373 1616 1 CALLING SEQUENCE:
1374 1617 1
1375 1618 1     ret_status.wlc.v = ERASE_LINE ( P_PBCB.rab.r)
1376 1619 1
1377 1620 1 FORMAL PARAMETERS:
1378 1621 1
1379 1622 1     P_PBCB.rab.r           Address of PBCB
1380 1623 1
1381 1624 1 IMPLICIT INPUTS:
1382 1625 1
1383 1626 1     NONE
1384 1627 1
1385 1628 1 IMPLICIT OUTPUTS:
1386 1629 1
1387 1630 1     NONE
1388 1631 1
1389 1632 1 COMPLETION STATUS:
1390 1633 1
1391 1634 1     $$$_NORMAL           Normal successful completion
1392 1635 1                          errors from SMG$$OUTPUT
1393 1636 1
1394 1637 1 SIDE EFFECTS:
1395 1638 1
1396 1639 1     NONE
1397 1640 1 --

```

SMGSMIN  
1-016

Minimal update calculation  
ERASE\_LINE - Erase to end-of-line

0 7  
16-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGMIN.B32;1

Page 45  
(20)

```
: 1399      1641  2 BEGIN
: 1400      1642  2
: 1401      1643  2 BIND
: 1402      1644  2
: 1403      1645  2      PBCB      = .P_PBCB      : BLOCK[,BYTE];
: 1404      1646  2
: 1405      1647  2 LOCAL
: 1406      1648  2
: 1407      1649  2      STATUS;      ! local status
```

```
: 1409      1650      2  !+
: 1410      1651      2  !- Create the appropriate escape sequence.
: 1411      1652      2  !-
: 1412      1653      2  !-
: 1413      1654      2  $SMG$GET_TERM_DATA(ERASE_TO_END_LINE);
: 1414      1655      2  !+
: 1415      1656      2  !- Output the escape sequence.
: 1416      1657      2  !-
: 1417      1658      2  !-
: 1418      1659      2  !-
: 1419      1660      2  IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
: 1420      1661      2  THEN BEGIN
: 1421      1662      2  STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
: 1422      1663      2  .PBCB[PBCB_A_CAP_BUFFER]);
: 1423      1664      2  IF NOT .STATUS THEN RETURN .STATUS;
: 1424      1665      2  END;
: 1425      1666      2  !-
: 1426      1667      2  RETURN SS$_NORMAL
: 1427      1668      2  !-
: 1428      1669      1  END;      ! Routine ERASE_LINE
```

000C 00000 ERASE_LINE:						
	5E		10 C2 00002	.WORD	Save R2,R3	: 1609
	52	04	AC D0 00005	SUBL2	#16, SP	
		00FC	C2 D5 00009	MOVL	P PBCB, R2	: 1645
			09 12 0000D	TSTL	252(R2)	: 1654
	53	0108	C2 9E 0000F	BNEQ	1\$	
			63 D4 00014	MOVAB	264(R2), R3	
			2C 11 00016	CLRL	(R3)	
		04	AE D4 00018	BRB	2\$	
		04	AE 9F 0001B	CLRL	INPUT_ARGS	
		0104	C2 DD 0001E	PUSHAB	INPUT_ARGS	
	53	0108	C2 9E 00022	PUSHL	260(R2)	
			53 DD 00027	MOVAB	264(R2), R3	
			C2 9F 00029	PUSHL	R3	
		0100	8F 3C 0002D	PUSHAB	256(R2)	
	10	AE	AE 9F 00033	MOVZWL	#473, 16(SP)	
			00FC	PUSHAB	16(SP)	
			C2 9F 00036	PUSHAB	252(R2)	
00000000G	00		06 FB 0003A	CALLS	#6, SMG\$GET_TERM_DATA	
	19		50 E9 00041	BLBC	STATUS, 4\$	
			63 D5 00044	TSTL	(R3)	: 1660
			12 13 00046	BEQL	3\$	
		0104	C2 DD 00048	PUSHL	260(R2)	: 1663
			63 DD 0004C	PUSHL	(R3)	: 1662
			52 DD 0004E	PUSHL	R2	
00000000G	00		03 FB 00050	CALLS	#3, SMG\$\$OUTPUT	
	03		50 E9 00057	BLBC	STATUS, 4\$	: 1664
	50		01 D0 0005A	MOVL	#1, R0	: 1667
			04 0005D	RET		: 1669

; Routine Size: 94 bytes, Routine Base: \_SMG\$CODE + 07F1



SMGSMIN  
1-016

Minimal update calculation  
ERASE\_LINE - Erase to end-of-line

D 7  
16-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGMIN.B32;1

Page 47  
(21)

SM  
1-

SMG\$MIN  
1-016

Minimal update calculation  
ERASE\_LINE - Erase to end-of-line

E 7  
16-Sep-1984 00:52:18  
14-Sep-1984 13:09:53

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGMIN.B32;1

Page 48  
(22)

: 1430  
: 1431  
1670 1 END  
1671 0 ELUDOM

# PSECT SUMMARY

Name	Bytes	Attributes
_SMG\$CODE	2127	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

# Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	9	0	581	00:01.0
\$255\$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1
\$255\$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1	469	46	9	38	00:00.5

# COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS:SMGMIN/OBJ=OBJ\$:SMGMIN MSRC\$:SMGMIN/UPDATE=(ENH\$:SMGMIN)

: Size: 2127 code + 0 data bytes  
: Run Time: 00:46.7  
: Elapsed Time: 02:19.9  
: Lines/CPU Min: 2145  
: Lexemes/CPU-Min: 15513  
: Memory Used: 293 pages  
: Compilation Complete



0359

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY